A benchmark method for the propositional modal logics K, KT, S4

Alain Heuerding*, Stefan Schwendimann*
Institut für Informatik und angewandte Mathematik,
Universität Bern, Switzerland

October 17, 1996

Abstract

A lot of methods have been proposed (and sometimes implemented) for proof search in the propositional modal logics K, KT, and S4.

It is difficult to compare the usefulness of these methods in practice, since mostly only the execution times for a few simple formulas have been published. We try to improve this unsatisfactory situation by presenting a set of benchmark formulas. Note that we do not just list formulas, but give a method that allows to compare different provers today and in the future.

As a starting point we give the results we obtained when we applied this benchmark method to the Logics Workbench (LWB). Moreover we hope that the discussion of postulates concerning benchmark tests for automated theorem provers help to obtain improved benchmark methods for other logics, too.

1 Introduction

A lot of methods have been proposed (and sometimes implemented) for proof search in the propositional modal logics K, KT, and S4. An incomplete list: tableaux and sequent calculi, embeddings in other logics, connection method, inverse method, logical frameworks.

It is difficult to compare the usefulness of these methods in practice. In most publications no or only a few execution times are listed. There are some exceptions like [1] (31 formulas for K, KT, S4) and [2] (38 formulas for S4). However, these formulas are already today too simple to serve as benchmarks. For example it takes about 0.002 seconds to solve the hardest problem of [1] for K, KT, S4 with the Logics Workbench (LWB). The formulas of [2] are harder, but also there it takes only about 0.035 seconds to solve the hardest formula with the LWB.

This situation is very unsatisfactory. Therefore we decided to collect a set of benchmark problems for the propositional modal logics K, KT, and S4.

In classical predicate logic, the often-cited collection of Pelletier [9] has been replaced by the TPTP library [11]. Although this library is considerably large, it is still common to choose a dozen of these formulas out of this library and publish the execution times for these formulas. Therefore we do not just give a list of formulas, but present a method that makes it possible to compare different provers. (The producers of TPTP plan a so-called benchmark suite for the next version that should allow the computation of a performance index for automated theorem provers for classical predicate logic; cp. [11].)

*Work supported by the Swiss National Science Foundation, 21-43197.95.

The selection of the hard benchmark formulas was guided by the following postulates (see the following section for a thorough discussion).

1. Provable as well as unprovable formulas.
2. Formulas of various structures.
3. Some of the benchmark formulas are hard enough for forth-coming provers.
4. For each formula the result is already known today.
5. Simple 'tricks' do not help to solve the problems.
6. Applying the benchmark test to a prover takes not too much time.
7. The results can be summarized.

These postulates lead us to the exclusive use of scalable formulas. We test for which parameters n a certain prover can decide whether the scalable formula $A(n)$ is provable or not in less than 100 seconds. The drawback of scalable formulas is their regular structure. If 'by chance' a prover 'recognizes' this structure, it can perform extremly well for a certain class of formulas, but still work very bad for slightly different formulas. We try to overcome this problem by hiding their structure with superfluous subformulas, and by presenting a sufficient number of different classes.

Collecting and constructing scalable formulas – always keeping the seven postulates in mind – is a very time-consuming task. Therefore the number of classes we propose is not as large as we would like it to be. The number of formulas is no longer a problem in the approach chosen in [3]. There formulas in some sort of nested conjunctive normal form were constructed using a random generator. It seems possible to choose length, number of clauses, and modal depth such that the resulting formulas are on the edge between 'satisfiable' and 'not satisfiable'. On the other hand this method has two drawbacks: the formulas all have a similar structure (cp. postulate 2), and – more serious – the correct result is not known beforehand (cp. postulate 4). In our opinion a method relying on such formulas cannot replace our method, but could serve as a complementary method.

At the end of this report we apply the benchmark method on the decision procedures integrated in the LWB. Of course it is possible to implement much faster decision procedures for K, KT, and S4, but hopefully these results serve as a starting point.

This paper contains all the information that is required in order to use the benchmark method. For convenience we offer the possibility to fetch some formulas and the programs that produce these formulas via WWW.

2 Postulates

In this section we discuss the seven postulates mentioned in the introduction in more detail. Note that they are widely applicable, not just for the logics for which we present benchmark formulas in this report.

Provable as well as unprovable formulas.

Often only provable formulas are considered as benchmark fomulas. We think that not provable formulas are as interesting as provable ones.

Moreover, algorithms like the Greedy procedure in classical propositional logic ([8], [10]) can recognize many unprovable formulas in a very short time, but they cannot show that a formula is provable. A benchmark method must also be applicable for such semi-decision procedures.

Formulas of various structures

It is clear that four or five formulas are not enough to obtain representative results about the performance of a prover. However, also a large number of formulas does not guarantee the quality of the benchmark set. For example it is tempting to use the embedding of IPC in S4 in order to obtain a list of benchmark formulas for S4 from a list of benchmark formulas for IPC, but all the resulting formulas look similar. An even more dubious method it the use of just one class of formulas, e.g. the pigeonhole formulas.

Things look of course different if one has a certain application in mind and develops a tuned prover for this application, but here we want to measure the overall performance.

Some of the benchmark formulas are hard enough for forth-coming provers.

It is not enough if the benchmark test is hard enough for today's provers. The test should still be applicable for much faster computers and improved search methods in the future. Thus the test must contain formulas that are far too hard for existing provers.

For each formula the result is already known today.

The correct result, i.e. 'provable' or 'unprovable', must be known already today. Therefore it must be possible to check the provability resp. non-provability of the formulas with logical methods, and not just with theorem provers. Random formulas do not fulfill this postulate.

Simple 'tricks' do not help.

The addition of a simple trick to the prover should not influence the results.

Assume for example that a benchmark set for K contains formulas without $\Box$ and without $\Diamond$. Such formulas are provable in K iff they are provable in classical propositional logic. If a prover checks at the beginning whether modal operators occur in the formula, then he can apply a fast decision procedure for classical propositional logic in order to solve the problem. Therefore our benchmark test contains no such formulas. For example the pigeonhole formula is disguised with some 'superfluous' $\Box$ and $\Diamond$. If a prover recognizes this disguise, or if he sees during the proof search that a subproblem is purely classical, then we think that this is no longer a simple trick.

Of course it is impossible to foresee all such tricks, but at least one should try.

Execution of the benchmark test takes not too much time.

Nobody wants to spend a lot of time to measure the performance of his prover. In particular, the required time should not depend on the prover or on the computer. Therefore some limits must be used, e.g. 'if the prover cannot solve the problem in n seconds, then stop'.

Results can be summarized.

A list of hundreds of numbers is not a satisfactory result, since it makes a comparison of several provers almost impossible. Therefore it must be possible to summarize the results. On the other hand the result should still show some properties of the prover. We think that in most cases it is sensible to give a list of numbers and not just one number in order to describe the performance of the prover. (Remember the semi-decision procedures already mentioned above.)

3 Benchmark method for K, KT, and $\mathsf{S4}$

3.1 Formulas

For each logic L, the formulas are divided into 9 classes of provable and 9 classes of unprovable formulas. The formulas in each class are parametrized by a number in $\mathbb{N}$. We call the jth formula in the ith class of provable resp. unprovable formulas $F_{\mathrm{p},i,j}$ resp. $F_{\mathrm{n},i,j}$. Thus $L \vdash F_{\mathrm{p},i,j}$ and $L \nvdash F_{\mathrm{n},i,j}$ for all $i \in \{1, \ldots, 9\}$, $j \in \mathbb{N}$.

Let $t(C)$ be the time in seconds the prover takes to decide whether or not the formula C is provable, i.e. $t : \mathrm{Fml} \to \mathbb{N} \cup \infty$. We compute for each $i \in \{1, \ldots, 9\}$ the numbers $n_{\mathrm{p},i} := \max\{j \mid t(F_{\mathrm{p},i,j}) < 100\ \mathit{seconds}\}$ and $n_{\mathrm{n},i} := \max\{j \mid t(F_{\mathrm{n},i,j}) < 100\ \mathit{seconds}\}$. Thus $F_{\mathrm{p},i,n_{\mathrm{p},i}+1}$ is the first formula in the ith class of provable formulas such that the prover cannot decide its provability in less than 100 seconds ($F_{\mathrm{n},i,n_{\mathrm{n},i}+1}$ analogous).

3.2 Timing

It is not possible to describe a timing procedure that is sensible for all provers (think e.g. of non-deterministic and parallel provers). Therefore it is important that everybody who applies the benchmark method describes exactly how the timing took place. See section 9 for an example.

If possible, then observe the following conditions in order to make the comparison of different provers easier.

- The timing starts after the start-up of the prover
- No conversions, e.g. in negation normal form, before the timing starts.
- The timing includes the construction of the data structure that contains the formula. This is especially important if you put additional information into the data structure before starting the 'real' proof search.
- Make your prover accessible in some way, in order to give people the possibility to check your results.

3.3 Presentation of the results

We propose the following form to present your results. With this information it should be possible to check the obtained results. See section 9 for an example of a filled in form.

1. Name of the prover. Short description of the methods used by the prover (or a reference).
2. Availability of the prover.
3. Short list of features your prover offers besides checking the provability of formulas (or a reference). Examples: Is the prover tuned for a certain application ? Has the correctness of the prover been verified ? Can one use one prover for many logics ? Is a proof resp. a counter-model generated ?
4. Hardware that was used for the benchmark test.
5. An example of the way you timed your prover.
6. Results, i.e. the numbers $n_{\mathrm{p},i}$ and $n_{\mathrm{n},i}$ for all $i \in \{1, \ldots, 9\}$.
7. Discussion of the results (optional).

3.4 What about the postulates ?

Our method largely satisfies the seven postulates:

1. There are provable as well as not provable formulas.
2. The formulas have various properties (number of variables, modal depth, ...) and origins.
3. Each class contains arbitrarily hard formulas.
4. It is clear which formulas are provable.
5. We tried ...
6. Applying our benchmark method is rather tiresome if one does not automate it, but the required time seems reasonable to us.
7. For each logic the result consists of a list of 18 numbers.

The main problems are the postulate 5, and the — at least for semi-decision procedures — relatively small number of classes.

4 Notation

Logics: CPC stands for classical propositional logic. K, KT, S4 are the modal propositional logics we consider. In order to avoid misunderstanding we list the corresponding properties of the accessibility relation in Kripke semantics: K: neither reflexive nor transitive, KT: reflexive, but not transitive, S4: reflexive and transitive.

Variables and formulas: $\text{Var} = \{p_0, p_1, p_2, \ldots\}$ is the set of the variables. We inductively define the set of the formulas Fml: $\top, \bot \in \text{Fml}$; $P \in \text{Var} \Rightarrow P \in \text{Fml}$; $A \in \text{Fml} \Rightarrow \Box A, \Diamond A, \neg A \in \text{Fml}$; $A, B \in \text{Fml} \Rightarrow A \wedge B, A \vee B, A \to B, A \leftrightarrow B \in \text{Fml}$.

Conventions: We use meta-variables (sometimes with subscripts) as follows: A, C for formulas, k, n for elements of $\{1, 2, \ldots\}$.
$\vee$ and $\wedge$ are left-associative. $\neg$, $\Box$, $\Diamond$ have the highest priority, followed by $\wedge$, $\vee$, $\to$, and $\leftrightarrow$. We omit the outermost brackets. Example: $p_0 \vee p_1 \vee p_2 \to (\neg p_2 \wedge p_1) \leftrightarrow p_0$ stands for the formula $((((p_0 \vee p_1) \vee p_2) \to ((\neg p_2) \wedge p_1)) \leftrightarrow p_0)$.

Iterations: $\Box^0 A$ stands for the formula A, $\Box^n A$ stands for the formula $\Box\Box^{n-1}A$. $\Diamond^0 A$ stands for the formula A, $\Diamond^n A$ stands for the formula $\Diamond\Diamond^{n-1}A$. $\bigwedge_{i=n_1,\ldots,n_2}(A_i)$ stands for the formula $A_{n_1} \wedge A_{n_1+1} \wedge \ldots \wedge A_{n_2}$ (if $n_1 \leq n_2$) resp. for the formula true (if $n_1 > n_2$). $\bigvee_{i=n_1,\ldots,n_2}(A_i)$ stands for the formula $A_{n_1} \vee A_{n_1+1} \vee \ldots \vee A_{n_2}$ (if $n_1 \leq n_2$) resp. for the formula false (if $n_1 > n_2$).

Equality: $A \equiv C$ means that the two formulas A and C are (syntactically) equal.

Substitution: If $P_1, \ldots, P_n$ are variables, then $A\{C_1/P_1, \ldots, C_n/P_n\}$ is the formula A with simultaneously substituted P_1 by C_1, ..., P_n by C_n. Example: $(p_0 \vee \Box(p_1 \to p_0))\{p_2/p_0, \Diamond p_1 \leftrightarrow p_0/p_1\} \equiv p_2 \vee \Box((\Diamond p_1 \leftrightarrow p_0) \to p_2)$.

Lists: $[x_1, x_2, \ldots, x_n]$ is the list with the elements x_1, x_2, ..., x_n. Note that $[1,2]$, $[2,1]$, $[1,1,2]$ are three different lists. $\bigvee_{y \in [x_1,\ldots,x_n]}(A_x)$ stands for $A_{x_1} \vee \ldots \vee A_{x_n}$.

Standard formulas:

$\text{D} :\equiv \Box p_0 \to \Diamond p_0$
$\text{D}_2 :\equiv \Diamond \mathsf{true}$
$\text{B} :\equiv p_0 \to \Box\Diamond p_0$
$\text{T} :\equiv \Box p_0 \to p_0$
$\text{A4} :\equiv \Box p_0 \to \Box\Box p_0$
$\text{A5} :\equiv \neg\Box p_0 \to \Box\neg\Box p_0$
$\text{H} :\equiv \Box(p_0 \vee p_1) \wedge \Box(\Box p_0 \vee p_1) \wedge \Box(p_0 \vee \Box p_1) \to \Box p_0 \vee \Box p_1$

$\mathrm{L} :\equiv \Box(p_0 \wedge \Box p_0 \to p_1) \vee \Box(p_1 \wedge \Box p_1 \to p_0)$
$\mathrm{L}^+ :\equiv \Box(\Box p_0 \to p_1) \vee \Box(\Box p_1 \to p_0)$
$\mathrm{Grz} :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to p_0$
$\mathrm{Grz}_1 :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to \Box p_0$
$\mathrm{Dum} :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to (\Diamond\Box p_0 \to p_0)$
$\mathrm{Dum}_1 :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to (\Diamond\Box p_0 \to \Box p_0)$
$\mathrm{Dum}_4 :\equiv \Box(\Box(p_0 \to \Box p_0) \to p_0) \to (\Diamond\Box p_0 \to p_0 \vee \Box p_0)$

Negation normal form: We inductively define the function nnf for formulas without $\leftrightarrow$.

- $\mathrm{nnf}(\mathsf{true}) :\equiv \mathsf{true}$, $\mathrm{nnf}(\mathsf{false}) :\equiv \mathsf{false}$, $\mathrm{nnf}(\neg\mathsf{true}) :\equiv \mathsf{false}$, $\mathrm{nnf}(\neg\mathsf{false}) :\equiv \mathsf{true}$
- $\mathrm{nnf}(P) :\equiv P$, $\mathrm{nnf}(\neg P) :\equiv \neg P$
- $\mathrm{nnf}(\Box A) :\equiv \Box\mathrm{nnf}(A)$, $\mathrm{nnf}(\Diamond A) :\equiv \Diamond\mathrm{nnf}(A)$
- $\mathrm{nnf}(A \wedge B) :\equiv \mathrm{nnf}(A) \wedge \mathrm{nnf}(B)$, $\mathrm{nnf}(A \vee B) :\equiv \mathrm{nnf}(A) \vee \mathrm{nnf}(B)$
- $\mathrm{nnf}(A \to B) :\equiv \mathrm{nnf}(\neg A \vee B)$
- $\mathrm{nnf}(\neg\Box A) :\equiv \mathrm{nnf}(\Diamond\neg A)$, $\mathrm{nnf}(\neg\Diamond A) :\equiv \mathrm{nnf}(\Box\neg A)$
- $\mathrm{nnf}(\neg(A \wedge B)) :\equiv \mathrm{nnf}(\neg A \vee \neg B)$, $\mathrm{nnf}(\neg(A \vee B)) :\equiv \mathrm{nnf}(\neg A \wedge \neg B)$
- $\mathrm{nnf}(\neg(A \to B)) :\equiv \mathrm{nnf}(A \wedge \neg B)$
- $\mathrm{nnf}(\neg\neg A) :\equiv \mathrm{nnf}(A)$

Length and modal depth: We inductively define the functions length and modaldepth.

- $\mathrm{length}(\mathsf{true}) := 1$, $\mathrm{length}(\mathsf{false}) := 1$, $\mathrm{length}(P) := 1$
- $\circ \in \{\Box, \Diamond, \neg\} \Rightarrow \mathrm{length}(\circ A) := \mathrm{length}(A) + 1$
- $\circ \in \{\wedge, \vee, \to, \leftrightarrow\} \Rightarrow \mathrm{length}(A \circ B) := \mathrm{length}(A) + \mathrm{length}(B) + 1$

- $\mathrm{modaldepth}(\mathsf{true}) := 0$, $\mathrm{modaldepth}(\mathsf{false}) := 0$, $\mathrm{modaldepth}(P) := 0$
- $\circ \in \{\Box, \Diamond\} \Rightarrow \mathrm{modaldepth}(\circ A) := \mathrm{modaldepth}(A) + 1$
- $\mathrm{modaldepth}(\neg A) := \mathrm{modaldepth}(A)$
- $\circ \in \{\wedge, \vee, \to, \leftrightarrow\} \Rightarrow \mathrm{modaldepth}(A \circ B) := \max(\mathrm{modaldepth}(A), \mathrm{modaldepth}(B))$

Integer division: The functions div and mod are defined as usual. Example: $14 \text{ div } 3 = 4$, $14 \bmod 3 = 2$.

5 Presentation of formulas

For each class of formulas we list in the following section:

1. Idea: Why is the formula provable resp. unprovable ?
2. Hiding: How is the formula 'hidden' in order to make the problem harder to solve ?
3. Characteristics: Modal depth, number of variables, ...
4. An inductive definition of the formulas.

Appendix I contains the first two formulas of each class, and the length, depth, modal depth, variables of the first six formulas of each class. If you generate the formulas by yourself, then please compute these values and compare them with the ones in appendix I. In this way you can make (relatively) sure that you consider the correct formulas. (See section 4 for the definitions of length and modal depth of a formula.)

Appendix II contains the listings of the procedures (written in the LWB programming language) we used to generate the formulas for the benchmark test in section 9. The formulas you can fetch were obtained with the help of these programs.

In order to avoid inconsistencies between the formulas in the sections 6, 7, 8 and the programs in appendix II, we implemented a Perl program that automatically converts the LaTeX definitions into LWB programs. We then compared the results of these generated LWB programs with the results of the LWB programs in appendix II.

6 Formulas for K

6.1 *k_branch_p*

Idea: The branching formula as defined in [4], plus a negation symbol in front and the additional subformula $\neg\Box^n p_{n \text{ div } 3+1}$ in order to make the formula provable. We assume $n < 100$.

Charateristics: $2n+3$ variables, modal depth $O(n)$.

$$\mathit{k\_branch\_p}(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \bigwedge_{i=0,\ldots,n}(\Box^i(\mathit{bdepth}(n) \wedge \mathit{det}(n) \wedge \mathit{branching}(n)))) \vee \neg\Box^n p_{n \text{ div } 3+1}$$

$$\mathit{bdepth}(n) :\equiv \bigwedge_{i=1,\ldots,n+1}(p_{100+i} \rightarrow p_{99+i})$$

$$\mathit{det}(n) :\equiv \bigwedge_{i=0,\ldots,n}(p_{100+i} \rightarrow (p_i \rightarrow \Box(p_{100+i} \rightarrow p_i)) \wedge (\neg p_i \rightarrow \Box(p_{100+i} \rightarrow \neg p_i)))$$

$$\mathit{branching}(n) :\equiv$$
$$\bigwedge_{i=0,\ldots,n-1}(p_{100+i} \wedge \neg p_{101+i} \rightarrow \Diamond(p_{101+i} \wedge \neg p_{102+i} \wedge p_{i+1}) \wedge \Diamond(p_{101+i} \wedge \neg p_{102+i} \wedge \neg p_{i+1}))$$

6.2 *k_branch_n*

Idea: The branching formula as defined in [4].

Characteristics: Every Kripke model that satisfies the pure branching formula must have an exponential number of worlds (cp. [4]). $2n+3$ variables, modal depth $n+1$. We assume $n < 100$.

$$\mathit{k\_branch\_n}(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \bigwedge_{i=0,\ldots,n}(\Box^i(\mathit{bdepth}(n) \wedge \mathit{det}(n) \wedge \mathit{branching}(n))))$$

bdepth, *det*, *branching* as in 6.1.

6.3 *k_d4_p*

Idea: $\mathsf{K} \vdash \mathrm{D} \wedge \mathrm{A4} \wedge \mathrm{B}\{\neg p_0/p_0\} \rightarrow \mathrm{T}$.

Hiding: The left hand side occurs with 1 to n $\Box$, the right hand side occurs just with n $\Box$ in front. $\Box^n\mathrm{T}$ is repeated n times. Additionally some superfluous instances, and the whole formula in negation normal form.

Characteristics: One variable, modal depth $n+3$, in negation normal form.

$$\mathit{k\_d4\_p}(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(\Box^n\mathrm{T} \vee \neg\Box^i\mathrm{D}_2 \vee \neg\Box^i\mathrm{A4} \vee \neg\Box^i\mathrm{A4}\{\Diamond p_0/p_0\} \vee \neg\Box^i\mathrm{B} \vee \neg\Box^i\mathrm{B}\{\neg p_0/p_0\}))$$

6.4 *k_d4_n*

Idea: A5 is not provable in K plus any instances of D, A4, and T.

Hiding: As in 6.3.

Characteristics: One variable, modal depth $n+3$, in negation normal form.

$$\mathit{k\_d4\_n}(n) :\equiv \mathrm{nnf}(\bigvee_{i=1,\ldots,n}(\Box^n(\Box p_0 \vee \Box\Diamond\neg p_0) \vee \neg\Box^i\mathrm{D}_2 \vee \neg\Box^i\mathrm{A4} \vee \neg\Box^i\mathrm{A4}\{\Diamond p_0/p_0\} \vee \neg\Box^i\mathrm{D}$$
$$\vee \neg\Box^i\mathrm{A4}\{\Diamond p_0 \rightarrow p_0/p_0\} \vee \neg\Box^i\mathrm{A4}\{\Box p_0 \rightarrow p_0/p_0\}))$$

6.5 *k_dum_p*

Idea: $\mathsf{K} \vdash \mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\} \to \mathrm{Dum}_1$.

Hiding: Some of the formulas on the left hand side of the implication occur with 1 to $n-1$ $\Box$ in front, the right hand side occurs just once with n div $2+1$ $\Box$ in front.

Characteristics: One variable, modal depth $n+2$ (for $n > 4$).

$$\begin{aligned} \mathit{k\_dum\_p}(n) :\equiv\ & \textstyle\bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i(\Box\mathrm{A4} \wedge \mathrm{Dum})) \wedge \neg\Box^{n \text{ div } 2+1}\mathrm{Dum}_1 \\ & \to \Diamond^{n \text{ div } 2+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\}) \\ & \quad \vee \textstyle\bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg(\Box\mathrm{A4} \wedge \mathrm{Dum})) \end{aligned}$$

6.6 *k_dum_n*

Idea: Dum is not provable in K plus any instances of Dum_4 and A4.

Hiding: As in 6.5.

Characteristics: One variable, modal depth $n+5$.

$$\begin{aligned} \mathit{k\_dum\_n}(n) :\equiv\ & \textstyle\bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i(\Box\mathrm{A4} \wedge \mathrm{Dum}_4)) \wedge \neg\Box^{n+1}\mathrm{Dum} \\ & \to \Diamond^{n+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum}_4 \wedge \mathrm{Dum}_4\{p_0 \to \Box p_0/p_0\}) \\ & \quad \vee \textstyle\bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg(\Box\mathrm{A4} \wedge \mathrm{Dum}_4)) \end{aligned}$$

6.7 *k_grz_p*

Idea: $\mathsf{K} \vdash \Box\mathrm{Grz} \wedge \mathrm{Grz}\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \to \mathrm{Grz}_1$, where $C()$ is defined as below.

Hiding: Many superfluous instances with three variables, and iterated $\Box$ inside the instances.

Characteristics: 3 (if $n < 5$) resp. 4 variables, modal depth 7 (if $n < 14$) resp. $n + 14$ div 4.

$$\begin{aligned} \mathit{k\_grz\_p}(n) :\equiv\ & \Box\mathrm{Grz}\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \\ & \to \mathrm{Grz}_1\{p_1/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \vee \mathrm{Grz}_1\{p_3/p_0\} \end{aligned}$$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}\{\mathit{l2}(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}\{\Box\mathit{l2}(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}\{\Box\mathit{l2}(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}\{\Box\mathit{l2}(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

$$\mathit{l2}(i) :\equiv \begin{cases} \mathsf{false} & i = 0 \\ \Box\mathit{l2}(i-1) \vee p_1 \vee p_2 \vee p_3 \vee p_4 & \text{otherwise} \end{cases}$$

$$C() :\equiv \Box(p_2 \to \Box p_2) \to p_2$$

6.8 *k_grz_n*

Idea: Grz is not provable in K plus instances of Grz_1.

Hiding: As in 6.7.

Characteristics: 3 (if $n < 5$) resp. 4 variables, modal depth 7 (if $n < 14$) resp. $n + 14$ div 4.

$$\begin{aligned} \mathit{k\_grz\_n}(n) :\equiv\ & \Box\mathrm{Grz}_1\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}_1\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \\ & \to \mathrm{Grz}\{p_1/p_0\} \vee \mathrm{Grz}\{p_2/p_0\} \vee \mathrm{Grz}\{p_3/p_0\} \end{aligned}$$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}_1\{\mathit{l2}(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}_1\{\Box\mathit{l2}(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}_1\{\Box\mathit{l2}(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}_1\{\Box\mathit{l2}(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

$\mathit{l2}$, C as in 6.7.

6.9 *k_lin_p*

Idea: $\mathsf{K} \vdash H2 \rightarrow \mathrm{L}$, where $H2$ is the instance of H defined below.

Hiding: Superfluous instances of $H2$ with other variables.

Characteristics: $n+1$ variables, modal depth 3.

$$\begin{aligned} k\_lin\_p(n) :\equiv\ & \bigvee_{i=1,\ldots,n \text{ div } 3}(\neg H2(p_i, p_{i+1})) \\ & \vee \mathrm{L}\{p_n/p_0, p_n/p_1\} \\ & \vee \bigvee_{i=n \text{ div } 3+1,\ldots,n}(\neg H2(p_i, p_{i+1})) \end{aligned}$$

$$H2(A,B) :\equiv \mathrm{H}\{A \wedge \Box A \wedge A \rightarrow B/p_0, \neg A \rightarrow \neg(\Box B \wedge B)/p_1\}$$

6.10 *k_lin_n*

Idea: L^+ is not provable in K plus any instances of L.

Characteristics: $n+1$ variables, modal depth 2 (if $n=1$) resp. 3.

$$\begin{aligned} k\_lin\_n(n) :\equiv\ & \bigvee_{i=1,\ldots,2n-2}(\neg \mathrm{L}\{\Diamond p_i/p_0, p_{i+1}/p_1\} \vee \neg \mathrm{L}\{p_i \rightarrow \Box p_{i+1}/p_0, p_{i+1}/p_1\}) \\ & \vee \mathrm{L}^+\{p_n/p_0, p_n/p_1\} \\ & \vee \bigvee_{i=2n,\ldots,4n-4}(\neg \mathrm{L}\{\Diamond p_i/p_0, p_{i+1}/p_1\} \vee \neg \mathrm{L}\{p_i \rightarrow \Box p_{i+1}/p_0, p_{i+1}/p_1\}) \end{aligned}$$

6.11 *k_path_p*

Idea: Without hiding, the formula would just be $\Box p_{\mathrm{path\_el}(1,n)} \vee \Diamond(\neg p_{\mathrm{path\_el}(1,n)} \wedge \Box p_{\mathrm{path\_el}(2,n)}) \vee \ldots \vee \Diamond(\neg p_{\mathrm{path\_el}(n-1,n)} \wedge \Box p_{\mathrm{path\_el}}(n,n)) \vee \Diamond^n p_{\mathrm{path\_el}(n,n)}$.

Hiding: The formula above is the path from the entry to the exit of a labyrinth described by *k_path_p*. It makes no difference whether one starts at the entry or at the exit.

Characteristics: In backward proof search, the path from the entry to the exit has to be found with backtracking, i.e. the function *path_el* must be reconstructed. 6 variables, modal depth n.

$$\begin{aligned} k\_path\_p(n) :\equiv\ & (\Box p_1 \vee \Box p_{path\_el(1,n)} \vee \Box p_3 \vee \Box p_5) \\ & \vee \bigvee_{i=1,\ldots,n; j=1,\ldots,6}(left\_to\_right(i,j,n) \vee right\_to\_left(i,j,n)) \\ & \vee (\Diamond^n \neg p_2 \vee \Diamond^n \neg p_4 \vee \Diamond^n \neg p_{path\_el(n,n)} \vee \Diamond^n \neg p_6) \end{aligned}$$

$$path\_el(l,n) :\equiv \begin{cases} 2 & l = 1 \\ \text{modg } (path\_el(l-1,n)+3, 6) & l > n \text{ div } 2 \\ \text{modg } (path\_el(l-1,n)+5, 6) & \text{otherwise} \end{cases}$$

$$left\_to\_right(l,k,n) :\equiv$$
$$\begin{cases} \mathsf{false} & l = n \\ \mathsf{false} & l \neq n, k \bmod 2 = 0, k \neq path\_el(l,n) \\ lists2fml(l,k,[path\_el(l+1,n)]) & l \neq n, k \bmod 2 = 0, k = path\_el(l,n) \\ lists2fml(l,k,[1,3,path\_el(l+1,n),5]) & l \neq n, k \bmod 2 \neq 0, k = path\_el(l,n) \\ lists2fml(l,k,delete(path\_el(l+1,n),1,3,5)) & \text{otherwise} \end{cases}$$

$$right\_to\_left(l,k,n) :\equiv \begin{cases} \mathsf{false} & l = 1 \\ \mathsf{false} & l \neq 1, k \bmod 2 = 1 \\ lists2fml\_back(l-1,k,delete(path\_el(l-1,n),2,4,6)) & \text{otherwise} \end{cases}$$

$$lists2fml(l,k,s) :\equiv \bigvee_{x \in s}(\Diamond^l(\neg p_k \wedge \Box p_x))$$

$$lists2fml\_back(l,k,s) :\equiv \bigvee_{x \in s}(\Diamond^l(\neg p_x \wedge \Box p_k))$$

$$delete(x,y1,y2,y3) :\equiv \begin{cases} [y2, y3] & x = y1 \\ [y1, y3] & x = y2 \\ [y1, y2] & x = y3 \\ [y1, y2, y3] & \text{otherwise} \end{cases}$$

$$modg(n_1,n_2) :\equiv \begin{cases} n_2 & n_1 \bmod n_2 = 0 \\ n_1 \bmod n_2 & \text{otherwise} \end{cases}$$

6.12 *k_path_n*

Idea: As in 6.11, but one piece of the path is missing (cp. the different definitions of *left_to_right* in 6.11 and 6.12).

Hiding: As in 6.11.

Characteristics: In backward proof search, the labyrinth is searched (using backtracking) for a path from the entry to the exit. 6 variables, modal depth $n+1$.

$$\begin{aligned}
&\mathit{k_path_n}(n) :\equiv \\
&\quad (\Box p_1 \vee \Box p_{path_el(1,n+1)} \vee \Box p_3 \vee \Box p_5) \\
&\quad \vee \bigvee_{i=1,\ldots,n+1;j=1,\ldots,6} (\mathit{left_to_right}(i,j,n+1) \vee \mathit{right_to_left}(i,j,n+1)) \\
&\quad \vee (\Diamond^{n+1}\neg p_2 \vee \Diamond^{n+1}\neg p_4 \vee \Diamond^{n+1}\neg p_{path_el(n+1,n+1)} \vee \Diamond^{n+1}\neg p_6)
\end{aligned}$$

path_el, *right_to_left*, *lists2fml*, *lists2fml_back*, *delete*, *modg* as in 6.11.

$$\mathit{left\_to\_right}(l,k,n) :\equiv$$
$$\begin{cases}
\mathsf{false} & l = n \\
\mathsf{false} & l \neq n, k \bmod 2 = 0, k \neq \mathit{path_el}(l,n) \\
\mathsf{false} & l \neq n, k \bmod 2 = 0, k = \mathit{path_el}(l,n), l = n \text{ div } 2 \\
\mathit{lists2fml}(l,k,[\mathit{path_el}(l+1,n)]) & l \neq n, k \bmod 2 = 0, k = \mathit{path_el}(l,n), l \neq n \text{ div } 2 \\
\mathit{lists2fml}(l,k,[1,3,5]) & l \neq n, k \bmod 2 \neq 0, k = \mathit{path_el}(l,n), l = n \text{ div } 2 \\
\mathit{lists2fml}(l,k,[1,3,\mathit{path_el}(l+1,n),5]) & l \neq n, k \bmod 2 \neq 0, k = \mathit{path_el}(l,n), l \neq n \text{ div } 2 \\
\mathit{lists2fml}(l,k,\mathit{delete}(\mathit{path_el}(l+1,n),1,3,5)) & \text{otherwise}
\end{cases}$$

6.13 *k_ph_p*

Idea: The pigeonhole formulas. We assume $n < 100$.

Hiding: Some $\Box$ and $\Diamond$.

Characteristics: Essentially a CPC problem. n^2+n variables, modal depth 1 (if $n=1$) resp. 2.

$$\mathit{k\_ph\_p}(n) :\equiv \Diamond \mathit{left}(n) \to \Diamond \mathit{right}(n)$$

$$\mathit{left}(n) :\equiv \bigwedge_{i=1,\ldots,n+1} (\bigvee_{j=1,\ldots,n} (l(i,j)))$$

$$\mathit{right}(n) :\equiv \bigvee_{j=1,\ldots,n;i_1=1,\ldots,n+1;i_2=i_1+1,\ldots,n+1} (l(i_1,j) \wedge l(i_2,j))$$

$$l(i,j) :\equiv \begin{cases} \Box p_{100i+j} & i < j \\ p_{100i+j} & \text{otherwise} \end{cases}$$

6.14 *k_ph_n*

Idea: The pigeonhole formulas, with one missing conjunct on the right hand side. We assume $n < 100$.

Hiding: Some $\Box$ and $\Diamond$.

Characteristics: Essentially a CPC problem. n^2+n variables, modal depth 1 (if $n=1$) resp. 2.

$$\mathit{k\_ph\_n}(n) :\equiv \Diamond \mathit{left}(n) \to \Diamond \mathit{right}(n)$$

left, l as in 6.13.

$$\mathit{right}(n) :\equiv \bigvee_{j=1,\ldots,n;i_1=1,\ldots,n+1;i_2=i_1+1,\ldots,n+1} (\mathit{l2}(n,i_1,j) \wedge \mathit{l2}(n,i_2,j))$$

$$\mathit{l2}(n,i,j) :\equiv \begin{cases} \neg l(i,j) & i = j, i = (2n) \text{ div } 3 + 1 \\ l(i,j) & \text{otherwise} \end{cases}$$

6.15 *k_poly_p*

Idea: The formula $(p_1 \leftrightarrow p_2) \vee (p_2 \leftrightarrow p_3) \vee \ldots \vee (p_{n-1} \leftrightarrow p_n) \vee (p_n \leftrightarrow p_1)$ says: If we have a polygon with n vertices, and all the vertices are either black or white, then two adjacent vertices have the same colour. If n is odd, then this formula is provable in CPC.

Hiding: Many $\Box$, $\Diamond$, and superfluous subformulas.

Characteristics: Essentially a CPC problem. About $4.5n$ variables, modal depth about $3n$.

$$k\_poly\_p(n) :\equiv \begin{cases} poly(3n+1) & n \bmod 2 = 0 \\ poly(3n) & n \bmod 2 = 1 \end{cases}$$

$$poly(n) :\equiv \Box^{n+1} \bigwedge_{i=1,\ldots,n+1}(p_i) \vee f(n,n) \vee \Box^{n+1} \bigwedge_{i=1,\ldots,n+1}(\neg p_{2i})$$

$$f(i,n) :\equiv \begin{cases} \mathsf{false} & i = 0 \\ \Diamond(f(i-1,n) \vee \Diamond^i(p_n \leftrightarrow p_1)) \vee \Box p_{i+2} & i = n \\ \Diamond(f(i-1,n) \vee \Diamond^i(p_i \leftrightarrow p_{i+1})) \vee \Box p_{i+2} & \text{otherwise} \end{cases}$$

6.16 *k_poly_n*

Idea: As in 6.15, but for an even number of vertices.

Hiding: Many $\Box$, $\Diamond$, and superfluous subformulas. The superfluous subformulas do not influence the non-provability.

Characteristics: Essentially a CPC problem. About $4.5n$ variables, modal depth about $3n$.

$$k\_poly\_n(n) :\equiv \begin{cases} poly(3n) & n \bmod 2 = 0 \\ poly(3n+1) & n \bmod 2 = 1 \end{cases}$$

$poly$, f as in 6.15.

6.17 *k_t4p_p*

Idea: $\mathsf{K} \vdash \mathrm{T}\{\Diamond p_0/p_0\} \wedge \Box\mathrm{T}\{\neg\Box\Diamond p_0/p_0\} \wedge \mathrm{A4}\{\Diamond p_0/p_0\} \wedge \Box(\Diamond\Box\Diamond p_0 \to (p_0 \to \Box p_0)) \to \Diamond\Box p_0 \vee \Diamond\Box\neg p_0$

Hiding: Superfluous subformulas ($\Diamond\neg p_3$, $\Diamond p_4$), superfluous instances of A4 and A5, nested $\Box$.

Characteristics: 4 variables, modal depth $n+5$. Partially in negation normal form.

$$k\_t4p\_p(n) :\equiv E(n) \vee \mathrm{nnf}(\neg C(n)) \vee \Diamond p_4$$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Diamond p_1) \wedge \Box(\Box\neg\Box\Diamond p_1 \to \neg\Box\Diamond p_0) \wedge (\Box\Diamond p_0 \to \Box\Box\Diamond p_1) & \\ \quad \wedge\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1) \wedge \Box\Diamond p_1)\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) \wedge \Box\mathrm{A4}\{\Diamond p_1/p_0\} & \text{otherwise} \end{cases}$$

$$E(i) :\equiv \begin{cases} \Diamond\Box p_0 & i = 0 \\ \Diamond\neg\mathrm{A4}\{\neg p_1/p_0\} \vee \Box E(i-1) \vee \Box\mathrm{A5}\{p_1/p_0\} & \text{otherwise} \end{cases}$$

6.18 *k_t4p_n*

Idea: $\Diamond\Box p_0$ is not provable in K plus any instances of T, A4, and $\Box(\Diamond\Box\Diamond p_0 \to (p_0 \to \Box p_0))$.

Hiding: As in 6.17.

Characteristics: 4 variables, modal depth $2n+4$. Partially in negation normal form.

$$k\_t4p\_n(n) :\equiv E(2n-1) \vee \mathrm{nnf}(\neg C(2n-1)) \vee \Diamond p_4$$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Diamond p_1) \wedge \Box(\Box\neg\Box\Diamond p_1 \to \neg\Box\Diamond p_0) \wedge (\Box\Diamond p_0 \to \Box\Box\Diamond p_1) & \\ \quad \wedge\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1))\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) \wedge \Box\mathrm{A4}\{\Diamond p_1/p_0\} & \text{otherwise} \end{cases}$$

E as in 6.17.

7 Formulas for KT

7.1 *kt_45_p*

Idea: $\mathsf{KT} \vdash \mathrm{A5}\{\Box p_0/p_0\} \wedge \Box\mathrm{A5}\{\neg p_0/p_0\} \to \mathrm{A4}$.

Hiding: The left hand side occurs with 1 to n $\Box$, the right hand side occurs just with n $\Box$ in front. Additionally some superfluous instances, and the whole formula in negation normal form.

Characteristics: One variable, modal depth $n+3$, in negation normal form.

$$\mathit{kt\_45\_p}(n) :\equiv \mathrm{nnf}(\textstyle\bigvee_{i=1,\ldots,n}(\Box^n \mathrm{A4} \vee \neg\Box^i \mathrm{D}_2 \vee \neg\Box^i \mathrm{A5}\{\Diamond\neg p_0/p_0\} \vee \neg\Box^i\Box\mathrm{A5} \vee \neg\Box^i \mathrm{B}))$$

7.2 *kt_45_n*

Idea: A5 is not provable in KT plus any instances of A4.

Hiding: As in 7.1.

Characteristics: One variable, modal depth $n+3$, in negation normal form.

$$\begin{aligned}\mathit{kt\_45\_n}(n) :\equiv \mathrm{nnf}(\textstyle\bigvee_{i=1,\ldots,n}&(\Box^n(\Box p_0 \vee \Box\Diamond\neg p_0) \vee \neg\Box^i\mathrm{A4} \vee \neg\Box^i\mathrm{A4}\{\Diamond p_0/p_0\} \vee \neg\Box^i\mathrm{T}\\ &\vee \neg\Box^i\mathrm{A4}\{\Diamond p_0 \to p_0/p_0\} \vee \neg\Box^i\mathrm{A4}\{\Box p_0 \to p_0/p_0\}))\end{aligned}$$

7.3 *kt_branch_p*

Idea: The branching formula as described in [4], plus a negation symbol in front and the additional subformula $\neg\Box^n p_{(n \text{ div } 3)+1}$ in order to make the formula provable.

Charateristics: $2n+3$ variables, modal depth $n+1$.

$$\mathit{kt\_branch\_p}(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box^n(\mathit{bdepth}(n) \wedge \mathit{det}(n) \wedge \mathit{branching}(n))) \vee \neg\Box^n p_{(n \text{ div } 3)+1}$$

bdepth, *det*, *branching* as in 6.1.

7.4 *kt_branch_n*

Idea: The branching formula as defined in [4].

Characteristics: Every Kripke model that satisfies the pure branching formula must have an exponential number of worlds (cp. [4]). $2n+3$ variables, modal depth $n+1$.

$$\mathit{kt\_branch\_n}(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box^n(\mathit{bdepth}(n) \wedge \mathit{det}(n) \wedge \mathit{branching}(n)))$$

bdepth, *det*, *branching* as in 6.1.

7.5 *kt_dum_p*

Idea: $\mathsf{KT} \vdash \mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\} \to \mathrm{Dum}_1$.

Hiding: Some of he formulas on the left hand side of the implication occur with 1 to $n-1$ $\Box$ in front, the right hand side occurs just once with $(n \text{ div } 2)+1$ $\Box$ in front.

Characteristics: One variable, modal depth $n+1$ (if $n > 6$).

$$\begin{aligned}\mathit{kt\_dum\_p}(n) :\equiv \textstyle\bigwedge_{i=1,\ldots,n \text{ div } 2}&(\Box^i\mathrm{A4}) \wedge \neg\Box^{(n \text{ div } 2)+1}\mathrm{Dum}_1\\ &\to \Diamond^{(n \text{ div } 2)+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\})\\ &\quad \vee \textstyle\bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg\mathrm{A4})\end{aligned}$$

7.6 *kt_dum_n*

Idea: Grz is not provable in KT plus any instances of Dum.

Hiding: As in 7.5.

Characteristics: One variable, modal depth $n+2$ (if $n>4$).

$$\begin{aligned} kt\_dum\_n(n) :\equiv {} & \textstyle\bigwedge_{i=1,\ldots,n \text{ div } 2}(\Box^i(\Box \mathrm{A4} \wedge \mathrm{Dum})) \wedge \neg\Box^{n \text{ div } 2+1}\mathrm{Grz} \\ & \to \Diamond^{n \text{ div } 2+1}\neg(\mathrm{A4}\{\Box(p_0 \to \Box p_0) \to p_0/p_0\} \wedge \Box\mathrm{A4} \wedge \mathrm{Dum} \wedge \mathrm{Dum}\{p_0 \to \Box p_0/p_0\}) \\ & \quad \vee \textstyle\bigvee_{i=n \text{ div } 2+2,\ldots,n-1}(\Diamond^i\neg(\Box\mathrm{A4} \wedge \mathrm{Dum})) \end{aligned}$$

7.7 *kt_grz_p*

Idea: $\mathsf{KT} \vdash \Box\mathrm{Grz} \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0\} \to \mathrm{Grz}_1$, where C is defined as below.

Hiding: Many superfluous instances with iterated $\Box$ inside the instances. The subformulas $\Diamond\Box\neg p_0$ and $\Diamond p_0$ are the parts of an instance of T.

Characteristics: 3 (if $n<5$) resp. 4 variables, modal depth 7 (if $n<14$) resp. $n+14$ div 4.

$$\begin{aligned} kt\_grz\_p(n) :\equiv {} & \Box\mathrm{Grz}\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0\} \\ & \to \mathrm{Grz}_1\{p_1/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \Diamond\Box\neg p_0 \vee \mathrm{Grz}_1\{p_3/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \Diamond p_0 \end{aligned}$$

l, $l2$, C as in 6.7.

7.8 *kt_grz_n*

Idea: A5 is not provable in KT plus instances of Grz_1.

Hiding: As in 6.7.

Characteristics: 3 (if $n<5$) resp. 4 variables, modal depth 7 (if $n<14$) resp. $n+14$ div 4.

$$\begin{aligned} kt\_grz\_n(n) :\equiv {} & \Box\mathrm{Grz}_1\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}_1\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \\ & \to \mathrm{A5}\{p_1/p_0\} \vee \mathrm{A5}\{p_2/p_0\} \vee \mathrm{A5}\{p_3/p_0\} \end{aligned}$$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}_1\{l2(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

$l2$, C as in 6.7.

7.9 *kt_md_p*

Idea: In backward proof search, we have to find the way to $g(n,n,\neg p_1)$ through a lot of $\Box$ and $\Diamond$.

Charateristics: One (if $n=1$) resp. two variables, modal depth 0 (if $n=1$) resp. n^2-n+1.

$$\begin{aligned} & kt\_md\_p(n) :\equiv \\ & \quad p_1 \vee \textstyle\bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_2)) \vee g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)) \vee g(i,n,\neg p_2 \wedge \Diamond f(1,n,p_1))) \\ & \quad \vee g(n,n,\neg p_1) \end{aligned}$$

$$g(i,n,A) :\equiv \begin{cases} A & i=1 \\ f(i,n,g(i-1,n,A)) & \text{otherwise} \end{cases}$$

$$f(i,n,A) :\equiv \Diamond^{i-1}\Box\Diamond^{n-i}A$$

7.10 *kt_md_n*

Idea: Similar to 7.9, but simpler. The subformula $g(n,n,\neg p_1)$ is missing in order to make the formulas unprovable.

Charateristics: One variable, modal depth 0 (if $n=1$) resp. n^2-n+1.

$$kt\_md\_n(n) :\equiv p_1 \vee \textstyle\bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)))$$

f, g as in 7.9.

7.11 *kt_path_p*

Idea, hiding: As in 6.11, but we use new variables on each level of the labyrinth. (The formulas *k_path_p* collapse in KT since $\mathsf{KT} \vdash \Diamond^n A \to A$.)

Characteristics: $6n$ variables, modal depth n.

$$\begin{aligned} \mathit{kt\_path\_p}(n) :\equiv\ & (\Box p_{11} \vee \Box p_{10+\mathit{path\_el}(1,n)} \vee \Box p_{13} \vee \Box p_{15}) \\ & \vee \textstyle\bigvee_{i=1,\ldots,n;j=1,\ldots,6}(\mathit{left\_to\_right}(i,j,n) \vee \mathit{right\_to\_left}(i,j,n)) \\ & \vee (\Diamond^n \neg p_{10n+2} \vee \Diamond^n \neg p_{10n+4} \vee \Diamond^n \neg p_{10n+\mathit{path\_el}(n,n)} \vee \Diamond^n \neg p_{10n+6}) \end{aligned}$$

path_el, *left_to_right*, *right_to_left*, *delete*, *modg* as in 6.11.

$\mathit{lists2fml}(l,k,s) :\equiv \bigvee_{x \in s}(\Diamond^l(\neg p_{10l+k} \wedge \Box p_{10(l+1)+x}))$

$\mathit{lists2fml\_back}(l,k,s) :\equiv \bigvee_{x \in s}(\Diamond^l(\neg p_{10l+x} \wedge \Box p_{10(l+1)+k}))$

7.12 *kt_path_n*

Idea, hiding: As in 6.12, but we use new variables on each level of the labyrinth. (The formulas *k_path_p* collapse in KT since $\mathsf{KT} \vdash \Diamond^n A \to A$.)

Characteristics: $6n$ variables, modal depth $n+1$.

$$\begin{aligned} & \mathit{kt\_path\_n}(n) :\equiv \\ & \quad (\Box p_{11} \vee \Box p_{10+\mathit{path\_el}(1,n+1)} \vee \Box p_{13} \vee \Box p_{15}) \\ & \quad \vee \textstyle\bigvee_{i=1,\ldots,n+1;j=1,\ldots,6}(\mathit{left\_to\_right}(i,j,n+1) \vee \mathit{right\_to\_left}(i,j,n+1)) \\ & \quad \vee (\Diamond^{n+1} \neg p_{10(n+1)+2} \vee \Diamond^{n+1} \neg p_{10(n+1)+4} \vee \Diamond^{n+1} \neg p_{10(n+1)+\mathit{path\_el}(n+1,n+1)} \vee \Diamond^{n+1} \neg p_{10(n+1)+6}) \end{aligned}$$

path_el, *left_to_right*, *right_to_left*, *delete*, *modg* as in 6.12.

lists2fml, *lists2fml_back* as in 7.11.

7.13 *kt_ph_p*

Idea, hiding: As in 6.13.

Characteristics: Essentially a CPC problem. n^2+n variables, modal depth 1 (if $n=1$) resp. 2.

$\mathit{kt\_ph\_p}(n) :\equiv \mathit{left}(n) \to \Diamond \mathit{right}(n)$

left, *right*, *l* as in 6.13.

7.14 *kt_ph_n*

Idea, hiding: As in 6.14.

Characteristics: Essentially a CPC problem. n^2+n variables, modal depth 1 (if $n=1$) resp. 2.

$\mathit{kt\_ph\_n}(n) :\equiv \mathit{left}(n) \to \Diamond \mathit{right}(n)$

left, *right*, *l*, *l2* as in 6.14.

7.15 *kt_poly_p*

Idea, hiding: As in 6.15.

Characteristics: Essentially a CPC problem. About $7.5n$ variables, modal depth about $5n$.

$$\mathit{kt\_poly\_p}(n) :\equiv \begin{cases} \mathit{poly}(5n+1) & n \bmod 2 = 0 \\ \mathit{poly}(5n) & n \bmod 2 = 1 \end{cases}$$

poly as in 6.15.

$$f(i,n) :\equiv \begin{cases} \mathsf{false} & i = 0 \\ \Diamond(f(i-1,n) \vee \Diamond^{i+2}(p_n \leftrightarrow p_1)) \vee \Box p_{i+2} & i = n \\ \Diamond(f(i-1,n) \vee \Diamond^{i+2}(p_i \leftrightarrow p_{i+1})) \vee \Box p_{i+2} & \text{otherwise} \end{cases}$$

7.16 *kt_poly_n*

Idea, hiding: As in 6.16.

Characteristics: Essentially a CPC problem. About $7.5n$ variables, modal depth about $5n$.

$$kt\_poly\_n(n) :\equiv \begin{cases} poly(3n) & n \bmod 2 = 0 \\ poly(3n+1) & n \bmod 2 = 1 \end{cases}$$

$poly$, f as in 7.15.

7.17 *kt_t4p_p*

Idea, hiding: As in 6.17.

Characteristics: 4 variables, modal depth $n+4$. Partially in negation normal form.

$$kt\_t4p\_p(n) :\equiv E(n) \vee \mathrm{nnf}(\neg C(n)) \vee \Diamond p_4$$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Box\Box\Diamond p_1) \wedge \Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1) \wedge \Box\Diamond p_1)\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$$

E as in 6.17.

7.18 *kt_t4p_n*

Idea, hiding: As in 6.18.

Characteristics: 4 variables, modal depth $2n+3$. Partially in negation normal form.

$$kt\_t4p\_n(n) :\equiv E(2n-1) \vee \mathrm{nnf}(\neg C(2n-1)) \vee \Diamond p_4$$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Diamond p_1) \wedge (\Box\Diamond p_0 \to \Box\Box\Diamond p_1) & \\ \quad \wedge\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1))\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \Box\mathrm{A4}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$$

E as in 6.17.

8 Formulas for S4

8.1 *s4_45_p*

Idea: $\mathsf{S4} \vdash \mathrm{A5}\{\Box p_0/p_0\} \wedge \Box\mathrm{A5}\{\neg p_0/p_0\} \to \mathrm{A5}$.

Hiding: The left hand side occurs with 1 to n $\Box$, the right hand side occurs just with n $\Box$ in front. Additionally some superfluous instances, and the whole formula in negation normal form.

Characteristics: One variable, modal depth $n+3$, in negation normal form.

$$s4\_45\_p(n) :\equiv \mathrm{nnf}(\textstyle\bigvee_{i=1,\ldots,n}(h(n,\mathrm{A5}) \vee \neg h(i,\mathrm{D}_2) \vee \neg h(i,\mathrm{A5}\{\Diamond\neg p_0/p_0\}) \vee \neg h(i,\Box\mathrm{A5}) \vee \neg h(i,\mathrm{B})))$$

$$h(i,A) :\equiv \begin{cases} A & i = 0 \\ \Box p_0 \vee \Box h(i-1,A) \vee \Box p_1 & \text{otherwise} \end{cases}$$

8.2 *s4_45_n*

Idea: A5 is not provable in S4.

Hiding: As in 8.1.

Characteristics: One variable, modal depth $n+3$, in negation normal form.

$$\begin{aligned} s4\_45\_n(n) :\equiv \mathrm{nnf}(\textstyle\bigvee_{i=1,\ldots,n}&(h(n,(\Box p_0 \vee \Box\Diamond\neg p_0)) \vee \neg h(i,\mathrm{A4}) \vee \neg h(i,\mathrm{A4}\{\Diamond p_0/p_0\}) \vee \neg h(i,\mathrm{T}) \\ &\vee \neg h(i,\mathrm{A4}\{\Box p_0 \to p_0/p_0\}) \vee \neg h(i,\mathrm{T}\{\Box p_0 \to p_0/p_0\}))) \end{aligned}$$

h as in 8.1.

8.3 *s4_branch_p*

Idea: The branching formula as described in [4], plus a negation symbol in front and the additional subformula $\Box p_{(n \text{ div } 3)+1}$ in order to make the formula provable.

Charateristics: $2n+3$ variables, modal depth 2.

$$s4\_branch\_p(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box(bdepth(n) \wedge det(n) \wedge branching(n))) \vee \neg\Box p_{(n \text{ div } 3)+1}$$

$bdepth$, det, $branching$ as in 6.1.

8.4 *s4_branch_n*

Idea: The branching formula as defined in [4].

Characteristics: Every Kripke model that satisfies the pure branching formula must have an exponential number of worlds (cp. [4]). $2n+3$ variables, modal depth 2.

$$s4\_branch\_n(n) :\equiv \neg(p_{100} \wedge \neg p_{101} \wedge \Box(bdepth(n) \wedge det(n) \wedge branching(n)))$$

$bdepth$, det, $branching$ as in 6.1.

8.5 *s4_grz_p*

Idea: $\mathsf{S4} \vdash \Box\mathrm{Grz} \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0 \ \rightarrow \mathrm{Grz}_1$, where C is defined as below.

Hiding: Many superfluous instances with iterated $\Box$ inside the instances. The subformulas $\neg\Box\Diamond p_0$ and $\neg\Diamond\neg(\Box\Diamond p_0 \vee p_1)$ are the parts of a weakened instance of *A4*.

Characteristics: 4 (if $n < 5$) resp. 5 variables, modal depth 7 (if $n < 14$) resp. $n + 14$ div 4.

$$\begin{aligned} &s4\_grz\_p(n) :\equiv \\ &\quad \Box\mathrm{Grz}\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}\{C() \wedge (\mathrm{A4}\{C()/p_0\})/p_0\} \\ &\quad \rightarrow \mathrm{Grz}_1\{p_1/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \neg\Box\Diamond p_0 \vee \mathrm{Grz}_1\{p_3/p_0\} \vee \mathrm{Grz}_1\{p_2/p_0\} \wedge \neg\Diamond\neg(\Box\Diamond p_0 \vee p_1) \end{aligned}$$

l, $l2$, C as in 6.7.

8.6 *s4_grz_n*

Idea: A5 is not provable in $\mathsf{S4}$ plus instances of Grz_1.

Hiding: As in 6.7.

Characteristics: 3 (if $n < 5$) resp. 4 variables, modal depth 7 (if $n < 14$) resp. $n + 14$ div 4.

$$\begin{aligned} s4\_grz\_n(n) :\equiv\ &\Box\mathrm{Grz}_1\{p_2/p_0\} \wedge \textstyle\bigwedge_{i=1,\ldots,n-1}(l(i)) \wedge \mathrm{Grz}_1\{C() \wedge \mathrm{A4}\{C()/p_0\}/p_0\} \\ &\rightarrow \mathrm{A5}\{p_1/p_0\} \vee \mathrm{A5}\{p_2/p_0\} \vee \mathrm{A5}\{p_3/p_0\} \end{aligned}$$

$$l(i) :\equiv \begin{cases} \mathrm{Grz}_1\{l2(i \text{ div } 4)/p_0\} & i \bmod 4 = 0 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1/p_0\} & i \bmod 4 = 1 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2/p_0\} & i \bmod 4 = 2 \\ \mathrm{Grz}_1\{\Box l2(i \text{ div } 4) \vee p_1 \vee p_2 \vee p_3/p_0\} & \text{otherwise} \end{cases}$$

$l2$, C as in 6.7.

8.7 *s4_ipc_p*

Idea: We embed a formula that is provable in intuitionistic propositional logic in $\mathsf{S4}$.

Characteristics: n variables, modal depth 3.

$$s4\_ipc\_p(n) :\equiv \textstyle\bigwedge_{i=1,\ldots,n}(f(i,n)) \rightarrow \mathsf{false}$$

$$f(i,n) :\equiv \Box(\Box(\Box p_i \rightarrow \textstyle\bigwedge_{j=1,\ldots,n}(\Box p_j)) \rightarrow \mathsf{false})$$

8.8 *s4_ipc_n*

Idea: We embed a formula that is not provable in intuitionistic propositional logic in S4.
Characteristics: n variables, modal depth 3.

$s4\_ipc\_n(n) :\equiv \bigwedge_{i=1,\ldots,n}(f2(i,n)) \rightarrow \mathsf{false}$

$f2(i,n) :\equiv \begin{cases} \mathsf{true} & i = (n+1) \text{ div } 2 \\ f(i,n) & \text{otherwise} \end{cases}$

f as in 8.7.

8.9 *s4_md_p*

Idea: In backward proof search, we have to find the way to $g(n,n,\neg p_1)$ through a lot of $\Box$ and $\Diamond$.
Charateristics: One (if $n=1$) resp. two variables, modal depth 0 (if $n=1$) resp. n^2-n+1.

$s4\_md\_p(n) :\equiv$
$\quad p_1$
$\quad \vee \bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_2)) \vee g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)) \vee g(i,n,\neg p_2 \wedge \Diamond f(1,n,p_1)))$
$\quad \vee g(n,n,\neg p_1)$

g as in 7.9.

$f(i,n,A) :\equiv h(i-1, \Box h(n-i, A))$

$h(i,A) :\equiv \begin{cases} A & i = 0 \\ \Diamond h(i-1,A) \vee p_2 & \text{otherwise} \end{cases}$

8.10 *s4_md_n*

Idea: Similar to 8.9, but simpler. The subformula $g(n,n,\neg p_1)$ is missing in order to make the formulas unprovable.
Charateristics: One (if $n=1$) resp. two variables, modal depth 0 (if $n=1$) resp. n^2-n+1.

$s4\_md\_n(n) :\equiv p_1 \vee \bigvee_{i=1,\ldots,n-1}(g(i,n,\neg p_1 \wedge \Diamond f(1,n,p_1)))$

f, g, h as in 8.9.

8.11 *s4_path_p*

Idea, hiding: As in 7.11.
Characteristics: $6n$ variables, modal depth $n+1$.

$s4\_path\_p(n) :\equiv (\Box\Box p_{11} \vee \Box\Box p_{10+path\_el(1,n)} \vee \Box\Box p_{13} \vee \Box\Box p_{15})$
$\qquad \vee \bigvee_{i=1,\ldots,n;j=1,\ldots,6}(left\_to\_right(i,j,n) \vee right\_to\_left(i,j,n))$
$\qquad \vee \Diamond(\Diamond\neg p_{10n+2} \vee \Diamond\neg p_{10n+4} \vee \Diamond\neg p_{10n+path\_el(n,n)} \vee \Diamond\neg p_{10n+6})$

path_el, *left_to_right*, *right_to_left*, *delete*, *modg*, *lists2fml*, *lists2fml_back* as in 7.11.

8.12 *s4_path_n*

Idea, hiding: As in 7.12.
Characteristics: $6n$ variables, modal depth $n+2$.

$s4\_path\_n(n) :\equiv$
$\quad (\Box\Box p_{11} \vee \Box\Box p_{10+path\_el(1,n+1)} \vee \Box\Box p_{13} \vee \Box\Box p_{15})$
$\quad \vee \bigvee_{i=1,\ldots,n+1;j=1,\ldots,6}(left\_to\_right(i,j,n+1) \vee right\_to\_left(i,j,n+1))$
$\quad \vee \Diamond(\Diamond\neg p_{10(n+1)+2} \vee \Diamond\neg p_{10(n+1)+4} \vee \Diamond\neg p_{10(n+1)+path\_el((n+1),(n+1))} \vee \Diamond\neg p_{10(n+1)+6})$

path_el, *left_to_right*, *right_to_left*, *delete*, *modg*, *lists2fml*, *lists2fml_back* as in 7.12.

8.13 *s4_ph_p*

Idea: The pigeonhole formulas. We assume $n < 100$.

Hiding: Some $\Box$ and $\Diamond$.

Characteristics: Essentially a CPC problem. $O(n^2)$ variables, modal depth 2.

$s4\_ph\_p(n) :\equiv \mathit{left}(n) \to \Diamond \mathit{right}(n)$

$\mathit{right}(n) :\equiv \bigvee_{j=1,\ldots,n; i_1=1,\ldots,n+1; i_2=i_1+1,\ldots,n+1} (\Diamond(l(i_1, j) \wedge l(i_2, j)))$

left, l as in 6.13.

8.14 *s4_ph_n*

Idea: The pigeonhole formulas, with one missing conjunct on the right hand side. We assume $n < 100$.

Hiding: Some $\Box$ and $\Diamond$.

Characteristics: Essentially a CPC problem. $O(n^2)$ variables, modal depth 2.

$s4\_ph\_n(n) :\equiv \mathit{left}(n) \to \Diamond \mathit{right}(n)$

$\mathit{right}(n) :\equiv \bigvee_{j=1,\ldots,n; i_1=1,\ldots,n+1; i_2=i_1+1,\ldots,n+1} (\Diamond(l2(n, i_1, j) \wedge l2(n, i_2, j)))$

left, l as in 6.13.

$l2(n, i, j) :\equiv \begin{cases} \neg l(i, j) & i = j, i = (2n) \text{ div } 3 + 1 \\ l(i, j) & \text{otherwise} \end{cases}$

8.15 *s4_s5_p*

Idea: A formula that is provable in S5 embedded in S4.

Hiding: Some superfluous subformulas.

Characteristics: $3n$ variables, modal depth $3n$.

$s4\_s5\_p(n) :\equiv \Box\Diamond(\Box \bigvee_{i=1,\ldots,3n-2} (p_i \leftrightarrow p_{i+1}) \vee \Box p_{3n} \vee f(1, 3n-1)) \vee \Box(\Diamond p_1 \to \neg p_{3n})$

$f(i, n) :\equiv \begin{cases} \mathsf{false} & i = n \\ \Diamond(p_i \wedge \neg p_{i+1} \vee \neg p_i \wedge p_{i+1}) \vee \Box f(i+1, n) & \text{otherwise} \end{cases}$

8.16 *s4_s5_n*

Idea: A formula that is not provable in S5 embedded in S4.

Hiding: Some superfluous subformulas.

Characteristics: $6n$ variables, modal depth $6 - 4n$.

$s4\_s5\_n(n) :\equiv \Box\Diamond(\Box p_{6n} \vee f(1, 6n-5)) \vee \Box(\Diamond p_1 \to \neg p_{6n})$

f as in 8.15.

8.17 *s4_t4p_p*

Idea, hiding: As in 6.17.

Characteristics: 4 variables, modal depth $n + 4$. Partially in negation normal form.

$s4\_t4p\_p(n) :\equiv E(n) \vee \mathrm{nnf}(\neg C(2n-1)) \vee \Diamond p_4$

$C(i) :\equiv \begin{cases} (\Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1) \wedge \Box\Diamond p_1)\{p_0 \wedge \Diamond\neg p_3/p_1\} & i = 0 \\ \mathrm{Dum}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$

E as in 6.17.

8.18 $s4\_t4p\_n$

Idea, hiding: As in 6.18.

Characteristics: 4 variables, modal depth $2n+3$. Partially in negation normal form.

$s4\_t4p\_n(n) :\equiv E(2n-1) \vee \mathrm{nnf}(\neg C(4n-1)) \vee \Diamond p_4$

$$C(i) :\equiv \begin{cases} ((\Box\Diamond p_0 \to \Diamond p_1) \wedge \Box(\Diamond\Box\Diamond p_0 \wedge p_0 \to \Box p_1))\{p_0 \wedge \Diamond\neg p_3/p_1\} & i=0 \\ \mathrm{Dum}\{p_1/p_0\} \wedge \Box C(i-1) & \text{otherwise} \end{cases}$$

E as in 6.17.

9 Benchmark results for the LWB

Prover: We used the Logics Workbench (LWB), version 1.0. See [5] or the LWB home page `http://lwbwww.unibe.ch:8080/LWBinfo.html` for more information.

For all the logics the LWB uses backward proof search in two-sided sequent calculi. Use-check (similar to [6]) helps to cut off superfluous branches. In the case of S4, a loop-check is used (see [7]).

Programming language: C++ . Compiler: Sun C++ 4.0.1 . Operating system: Solaris 2.4 .

Availability: The binaries of the LWB 1.0 are available via the LWB home page (choose `about the LWB`, `install the LWB`).

You can also use the LWB 1.0 via WWW. Choose `run a session via WWW` on the LWB home page and type in your request.

Additional facilities of the prover:

- graphical user interface
- built-in programming language
- progress indicator: a slider shows how the proof search is going on
- various functions to convert formulas

Hardware: Sun SPARCstation 5, main memory: 80MB, 1 CPU (70 MHz microSPARC II)

Timing: The timing includes the parsing of the formulas and the construction of the corresponding data structure. The files loaded by the LWB have the following form:

```
load(k);
timestart; provable(box p0 -> box box p0); timestop;
quit;
```

Results:

class	$n_{p,i}$	class	$n_{n,i}$
k_branch_p	6	*k_branch_n*	7
k_d4_p	8	*k_d4_n*	6
k_dum_p	13	*k_dum_n*	19
k_grz_p	7	*k_grz_n*	13
k_lin_p	11	*k_lin_n*	8
k_path_p	12	*k_path_n*	10
k_ph_p	4	*k_ph_n*	8
k_poly_p	8	*k_poly_n*	11
k_t4p_p	8	*k_t4p_n*	7

class	$n_{p,i}$	class	$n_{n,i}$
kt_45_p	5	*kt_45_n*	4
kt_branch_p	5	*kt_branch_n*	6
kt_dum_p	5	*kt_dum_n*	10
kt_grz_p	6	*kt_grz_n*	> 20
kt_md_p	5	*kt_md_n*	5
kt_path_p	10	*kt_path_n*	9
kt_ph_p	4	*kt_ph_n*	8
kt_poly_p	14	*kt_poly_n*	2
kt_t4p_p	5	*kt_t4p_n*	7

class	$n_{\mathrm{p},i}$	class	$n_{\mathrm{n},i}$
s4_45_p	3	*s4_45_n*	5
s4_branch_p	11	*s4_branch_n*	7
s4_grz_p	9	*s4_grz_n*	> 20
s4_ipc_p	8	*s4_ipc_n*	7
s4_md_p	8	*s4_md_n*	6
s4_path_p	8	*s4_path_n*	6
s4_ph_p	4	*s4_ph_n*	8
s4_s5_p	4	*s4_s5_n*	9
s4_t4p_p	9	*s4_t4p_n*	12

In order to make absolutely clear how we obtained the numbers in the tables above from the run times, we give the run times for some of the formulas in the classes *k_branch_p* and *k_branch_n*. *k_branch_p*(6) resp. *k_branch_n*(7) are the last formulas that could be decided in less than 100 seconds; therefore we enter the numbers 6 resp. 7 in the first line of the table for K.

formula	run time (in seconds)
k_branch_p(1)	0.02
k_branch_p(2)	0.06
k_branch_p(3)	0.28
k_branch_p(4)	1.80
k_branch_p(5)	11.89
k_branch_p(6)	74.64
k_branch_p(7)	503.94

formula	run time (in seconds)
k_branch_n(1)	0.02
k_branch_n(2)	0.05
k_branch_n(3)	0.15
k_branch_n(4)	0.51
k_branch_n(5)	2.17
k_branch_n(6)	9.69
k_branch_n(7)	43.42
k_branch_n(8)	203.81

10 Availability of the formulas

You can get the first 15 formulas of each class as well as the LWB programs that generated these formulas. Load the LWB home page `http://lwbwww.unibe.ch:8080/LWBinfo.html` in a WWW browser and then choose the item `benchmarks`.

For each class of the three logics K, KT, S4 there is a compressed ASCII file (inside a tar file) that contains the first 15 formulas. The files contain a header line, a line with the word `begin` , then one formula on each line (with the corresponding number in front), and finally a line with the word `end` . The formulas are written in infix notation. The connectives are `~` for $\neg$, `&` for $\wedge$, `v` for $\vee$, `->` for $\rightarrow$, `<->` for $\leftrightarrow$. No brackets are omitted in the formulas in order to make the conversion into other formats easier.

It can happen that for some classes you need more than 15 formulas. Then you have several possibilities:

- Get the LWB procedures we used to generate the formulas, install the LWB, and generate the required formulas.
- Write a procedure that generates the formulas in this class according to the definitions in the sections 6, 7, 8. Please make sure that you generate the same formulas as we do by comparing the first 15 formulas, or at least by comparing the values of appendix I.
- Get the LWB procedures and use the LWB via WWW (choose the item `run a session` on the LWB home page). You have to replace the `read` statements in the files by the corresponding files for this purpose. Use `set("bracketmode", "full");` to obtain all the brackets.

11 Conclusion

We stated postulates for benchmark methods for proof search procedures. and presented a benchmark method for the propositional modal logics K, KT, S4 that satisfies these postulates.

Until now often just a few arbitrary — and often very easy — formulas were used to judge the efficiency of such theorem provers. With our method it is now possible to compare these provers in a standardized and fair way. Since we use but scalable formulas, the method will still be applicable when there are much faster machines and more efficient proof search procedures.

Of course the postulates are not limited to benchmark tests for modal logic theorem provers, but seem also reasonable for other non-classical logics.

References

[1] L. Catach. Tableaux: A general theorem prover for modal logics. *Journal of Automated Reasoning*, 7:489–510, 1991.

[2] S. Demri. Uniform and non uniform strategies for tableaux calculi for modal logics. *Journal of Applied Non-Classical Logics*, 5(1):77–96, 1995.

[3] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures – the case study of modal K. In *CADE 96*, LNCS, 1996.

[4] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[5] A. Heuerding, G. Jäger, S. Schwendimann, and M. Seyfried. Propositional logics on the computer. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Tableaux 95*, LNCS 918, pages 310–323, 1995.

[6] A. Heuerding and S. Schwendimann. On the modal logic K plus theories. In H. Kleine Büning, editor, *CSL 95*, LNCS 1092, pages 308–319, 1996.

[7] A. Heuerding, M. Seyfried, and H. Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Tableaux 96*, LNCS 1071, pages 210–225, 1996.

[8] E. Koutsoupias and C. Papadimitriou. On the greedy algorithm for satisfiability. *Information Processing Letters*, 43:53–55, 1992.

[9] F. Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2:191–216, 1986.

[10] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings 10th AAAI*, pages 47–59, 1992.

[11] G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In A. Bundy, editor, *CADE 12*, LNCS 814, pages 252–266, 1994.

12 Appendix I: Formulas generated with the LWB

12.1 K

12.1.1 k_branch_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_branch_p(i)), ", ", modaldepth(k_branch_p(i)), ", ", vars(k_branch_p(i)));
  1:  142, 2, [p0, p1, p100, p101, p102]
  2:  343, 3, [p0, p1, p2, p100, p101, p102, p103]
  3:  633, 4, [p0, p1, p2, p3, p100, p101, p102, p103, p104]
  4:  1012, 5, [p0, p1, p2, p3, p4, p100, p101, p102, p103, p104, p105]
  5:  1480, 6, [p0, p1, p2, p3, p4, p5, p100, p101, p102, p103, p104, p105, p106]
  6:  2037, 7, [p0, p1, p2, p3, p4, p5, p6, p100, p101, p102, p103, p104, p105, p106, p107]
> k_branch_p(1);
  ~(p100 & ~p101 & ((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101 ->
(p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & box
((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101 -> (p1 -> box (p101 ->
p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1))))) v ~box p1
> k_branch_p(2);
  ~(p100 & ~p101 & ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 ->
~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102 ->
~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia (p102
& ~p103 & ~p2))) & box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 ->
~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102 ->
~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia (p102
& ~p103 & ~p2)))) & box box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100
-> ~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102
-> ~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia
(p102 & ~p103 & ~p2)))))) v ~box box p1
```

12.1.2 k_branch_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_branch_n(i)), ", ", modaldepth(k_branch_n(i)), ", ", vars(k_branch_n(i)));
  1:  138, 2, [p0, p1, p100, p101, p102]
  2:  338, 3, [p0, p1, p2, p100, p101, p102, p103]
  3:  627, 4, [p0, p1, p2, p3, p100, p101, p102, p103, p104]
  4:  1005, 5, [p0, p1, p2, p3, p4, p100, p101, p102, p103, p104, p105]
  5:  1472, 6, [p0, p1, p2, p3, p4, p5, p100, p101, p102, p103, p104, p105, p106]
  6:  2028, 7, [p0, p1, p2, p3, p4, p5, p6, p100, p101, p102, p103, p104, p105, p106, p107]
> k_branch_n(1);
  ~(p100 & ~p101 & ((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101 ->
(p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & box
((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101 -> (p1 -> box (p101 ->
p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)))))
> k_branch_n(2);
  ~(p100 & ~p101 & ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 ->
~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102 ->
~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia (p102
& ~p103 & ~p2))) & box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 ->
~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102 ->
~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia (p102
& ~p103 & ~p2)))) & box box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100
-> ~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102
-> ~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia
(p102 & ~p103 & ~p2))))))
```

12.1.3 k_d4_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_d4_p(i)), ", ", modaldepth(k_d4_p(i)), ", ", vars(k_d4_p(i)));
  1:  46, 4, [p0]
  2:  100, 5, [p0]
  3:  161, 6, [p0]
  4:  229, 7, [p0]
  5:  304, 8, [p0]
  6:  386, 9, [p0]
> k_d4_p(1);
  box (dia ~p0 v p0) v dia box false v dia (box p0 & dia dia ~p0) v dia (box dia p0 & dia dia box ~p0) v dia (p0 & dia box ~p0) v
dia (~p0 & dia box p0)
> k_d4_p(2);
  box box (dia ~p0 v p0) v dia box false v dia (box p0 & dia dia ~p0) v dia (box dia p0 & dia dia box ~p0) v dia (p0 & dia box ~p0)
v dia (~p0 & dia box p0) v (box box (dia ~p0 v p0) v dia dia box false v dia dia (box p0 & dia dia ~p0) v dia dia (box dia p0 & dia
dia box ~p0) v dia dia (p0 & dia box ~p0) v dia dia (~p0 & dia box p0))
```

12.1.4 k_d4_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_d4_n(i)), ", ", modaldepth(k_d4_n(i)), ", ", vars(k_d4_n(i)));
  1:  72, 4, [p0]
  2:  153, 5, [p0]
  3:  242, 6, [p0]
  4:  339, 7, [p0]
  5:  444, 8, [p0]
  6:  557, 9, [p0]
> k_d4_n(1);
  box (box p0 v box dia ~p0) v dia box false v dia (box p0 & dia dia ~p0) v dia (box dia p0 & dia dia box ~p0) v dia (box p0 & box
~p0) v dia (box (box ~p0 v p0) & dia dia (dia p0 & ~p0)) v dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0))
> k_d4_n(2);
  box box (box p0 v box dia ~p0) v dia box false v dia (box p0 & dia dia ~p0) v dia (box dia p0 & dia dia box ~p0) v dia (box p0
& box ~p0) v dia (box (box ~p0 v p0) & dia dia (dia p0 & ~p0)) v dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0)) v (box box (box
p0 v box dia ~p0) v dia dia box false v dia dia (box p0 & dia dia ~p0) v dia dia (box dia p0 & dia dia box ~p0) v dia dia (box p0 &
box ~p0) v dia dia (box (box ~p0 v p0) & dia dia (dia p0 & ~p0)) v dia dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0)))
```

12.1.5 k_dum_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_dum_p(i)), ", ", modaldepth(k_dum_p(i)), ", ", vars(k_dum_p(i)));
  1:  95, 5, [p0]
  2:  119, 5, [p0]
  3:  119, 5, [p0]
  4:  146, 6, [p0]
  5:  172, 7, [p0]
  6:  201, 7, [p0]
> k_dum_p(1);
  true & ~box (box (box (box (p0 -> box p0) -> p0) -> dia box p0 -> box p0) -> dia ~((box (box (p0 -> box p0) -> p0) -> box box (box (p0
-> box p0) -> p0)) & box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0) & (box (box ((p0 -> box p0)
-> box (p0 -> box p0)) -> p0 -> box p0) -> dia box (p0 -> box p0) -> p0 -> box p0)) v false
> k_dum_p(2);
  box (box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0)) & ~box box (box (box (p0 -> box p0) ->
p0) -> dia box p0 -> box p0) -> dia dia ~((box (box (p0 -> box p0) -> p0) -> box box (box (p0 -> box p0) -> p0)) & box (box p0 -> box
box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0) & (box (box ((p0 -> box p0) -> box (p0 -> box p0)) -> p0 -> box p0)
-> dia box (p0 -> box p0) -> p0 -> box p0)) v false
```

12.1.6 k_dum_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_dum_n(i)), ", ", modaldepth(k_dum_n(i)), ", ", vars(k_dum_n(i)));
  1:  105, 6, [p0]
  2:  132, 7, [p0]
  3:  134, 8, [p0]
  4:  164, 9, [p0]
  5:  195, 10, [p0]
  6:  227, 11, [p0]
> k_dum_n(1);
  true & ~box box (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0) -> dia dia ~((box (box (p0 -> box p0) -> p0) -> box box (box
(p0 -> box p0) -> p0)) & box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0 v box p0) & (box (box ((p0
-> box p0) -> box (p0 -> box p0)) -> p0 -> box p0) -> dia box (p0 -> box p0) -> (p0 -> box p0) v box (p0 -> box p0))) v false
> k_dum_n(2);
  box (box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0 v box p0)) & ~box box box (box (box (p0 ->
box p0) -> p0) -> dia box p0 -> p0) -> dia dia dia ~((box (box (p0 -> box p0) -> p0) -> box box (box (p0 -> box p0) -> p0)) & box (box
p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0 v box p0) & (box (box ((p0 -> box p0) -> box (p0 -> box p0))
-> p0 -> box p0) -> dia box (p0 -> box p0) -> (p0 -> box p0) v box (p0 -> box p0))) v false
```

12.1.7 k_grz_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_grz_p(i)), ", ", modaldepth(k_grz_p(i)), ", ", vars(k_grz_p(i)));
  1:  160, 7, [p1, p2, p3]
  2:  181, 7, [p1, p2, p3]
  3:  212, 7, [p1, p2, p3]
  4:  251, 7, [p1, p2, p3]
  5:  298, 7, [p1, p2, p3, p4]
  6:  357, 7, [p1, p2, p3, p4]
> k_grz_p(1);
```

```
box (box (box (p2 -> box p2) -> p2) -> p2) & true & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (box (p1 -> box p1) -> p1) -> box p1) v (box (box (p2 -> box p2) -> p2) -> box p2) v (box (box (p3 -> box p3) -> p3) -> box p3)
> k_grz_p(2);
box (box (box (p2 -> box p2) -> p2) -> p2) & (box (box (box false v p1 -> box (box false v p1)) -> box false v p1) -> box false v p1) & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (box (p1 -> box p1) -> p1) -> box p1) v (box (box (p2 -> box p2) -> p2) -> box p2) v (box (box (p3 -> box p3) -> p3) -> box p3)
```

12.1.8 k_grz_n

```
> for i := 1 to 6 do
  print(i, ":  ", length(k_grz_n(i)), ", ", modaldepth(k_grz_n(i)), ", ", vars(k_grz_n(i)));
1:  159, 7, [p1, p2, p3]
2:  181, 7, [p1, p2, p3]
3:  213, 7, [p1, p2, p3]
4:  253, 7, [p1, p2, p3]
5:  301, 7, [p1, p2, p3, p4]
6:  361, 7, [p1, p2, p3, p4]
> k_grz_n(1);
box (box (box (p2 -> box p2) -> p2) -> box p2) & true & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (box (p1 -> box p1) -> p1) -> p1) v (box (box (p2 -> box p2) -> p2) -> p2) v (box (box (p3 -> box p3) -> p3) -> p3)
> k_grz_n(2);
box (box (box (p2 -> box p2) -> p2) -> box p2) & (box (box (box false v p1 -> box (box false v p1)) -> box false v p1) -> box (box false v p1)) & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (box (p1 -> box p1) -> p1) -> p1) v (box (box (p2 -> box p2) -> p2) -> p2) v (box (box (p3 -> box p3) -> p3) -> p3)
```

12.1.9 k_lin_p

```
> for i := 1 to 6 do
  print(i, ":  ", length(k_lin_p(i)), ", ", modaldepth(k_lin_p(i)), ", ", vars(k_lin_p(i)));
1:  97, 3, [p1, p2]
2:  177, 3, [p1, p2, p3]
3:  255, 3, [p1, p2, p3, p4]
4:  335, 3, [p1, p2, p3, p4, p5]
5:  415, 3, [p1, p2, p3, p4, p5, p6]
6:  495, 3, [p1, p2, p3, p4, p5, p6, p7]
> k_lin_p(1);
false v (box (p1 & box p1 -> p1) v box (p1 & box p1 -> p1)) v ~(box ((p1 & box p1 & p1 -> p2) v (~p1 -> ~(box p2 & p2))) & box (box (p1 & box p1 & p1 -> p2) v (~p1 -> ~(box p2 & p2))) & box ((p1 & box p1 & p1 -> p2) v box (~p1 -> ~(box p2 & p2))) -> box (p1 & box p1 & p1 -> p2) v box (~p1 -> ~(box p2 & p2)))
> k_lin_p(2);
false v (box (p2 & box p2 -> p2) v box (p2 & box p2 -> p2)) v (~(box ((p1 & box p1 & p1 -> p2) v (~p1 -> ~(box p2 & p2))) & box (box (p1 & box p1 & p1 -> p2) v (~p1 -> ~(box p2 & p2))) & box ((p1 & box p1 & p1 -> p2) v box (~p1 -> ~(box p2 & p2))) -> box (p1 & box p1 & p1 -> p2) v box (~p1 -> ~(box p2 & p2))) v ~(box ((p2 & box p2 & p2 -> p3) v (~p2 -> ~(box p3 & p3))) & box (box (p2 & box p2 & p2 -> p3) v (~p2 -> ~(box p3 & p3))) & box ((p2 & box p2 & p2 -> p3) v box (~p2 -> ~(box p3 & p3))) -> box (p2 & box p2 & p2 -> p3) v box (~p2 -> ~(box p3 & p3))))
```

12.1.10 k_lin_n

```
> for i := 1 to 6 do
  print(i, ":  ", length(k_lin_n(i)), ", ", modaldepth(k_lin_n(i)), ", ", vars(k_lin_n(i)));
1:  15, 2, [p1]
2:  149, 3, [p1, p2, p3, p4, p5]
3:  333, 3, [p1, p2, p3, p4, p5, p6, p7, p8, p9]
4:  517, 3, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13]
5:  701, 3, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17]
6:  885, 3, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21]
> k_lin_n(1);
false v (box (box p1 -> p1) v box (box p1 -> p1)) v false
> k_lin_n(2);
~(box (dia p1 & box dia p1 -> p2) v box (p2 & box p2 -> dia p1)) v ~(box ((p1 -> box p2) & box (p1 -> box p2) -> p2) v box (p2 & box p2 -> p1 -> box p2)) v (~(box (dia p2 & box dia p2 -> p3) v box (p3 & box p3 -> dia p2)) v ~(box ((p2 -> box p3) & box (p2 -> box p3) -> p3) v box (p3 & box p3 -> p2 -> box p3))) v (box (box p2 -> p2) v box (box p2 -> p2)) v (~(box (dia p4 & box dia p4 -> p5) v box (p5 & box p5 -> dia p4)) v ~(box ((p4 -> box p5) & box (p4 -> box p5) -> p5) v box (p5 & box p5 -> p4 -> box p5)))
```

12.1.11 k_path_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_path_p(i)), ", ", modaldepth(k_path_p(i)), ", ", vars(k_path_p(i)));
  1:  51, 1, [p1, p2, p3, p4, p5, p6]
  2:  156, 2, [p1, p2, p3, p4, p5, p6]
  3:  324, 3, [p1, p2, p3, p4, p5, p6]
  4:  455, 4, [p1, p2, p3, p4, p5, p6]
  5:  661, 5, [p1, p2, p3, p4, p5, p6]
  6:  818, 6, [p1, p2, p3, p4, p5, p6]
> k_path_p(1);
  box p1 v box p2 v box p3 v box p5 v (false v false v (false v false) v (false v false) v (false v false) v (false v false) v (false
v false)) v (dia ~p2 v dia ~p4 v dia ~p2 v dia ~p6)
> k_path_p(2);
  box p1 v box p2 v box p3 v box p5 v (dia (~p1 & box p1) v dia (~p1 & box p3) v false v (dia (~p2 & box p5) v false) v (dia (~p3
& box p1) v dia (~p3 & box p3) v false) v (false v false) v (dia (~p5 & box p1) v dia (~p5 & box p3) v false) v (false v false) v (false
v false) v (false v (dia (~p4 & box p2) v dia (~p6 & box p2))) v (false v false) v (false v (dia (~p4 & box p4) v dia (~p6 & box p4)))
v (false v false) v (false v (dia (~p4 & box p6) v dia (~p6 & box p6)))) v (dia dia ~p2 v dia dia ~p4 v dia dia ~p5 v dia dia ~p6)
```

12.1.12 k_path_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_path_n(i)), ", ", modaldepth(k_path_n(i)), ", ", vars(k_path_n(i)));
  1:  151, 2, [p1, p2, p3, p4, p5, p6]
  2:  319, 3, [p1, p2, p3, p4, p5, p6]
  3:  447, 4, [p1, p2, p3, p4, p5, p6]
  4:  653, 5, [p1, p2, p3, p4, p5, p6]
  5:  811, 6, [p1, p2, p3, p4, p5, p6]
  6:  1055, 7, [p1, p2, p3, p4, p5, p6]
> k_path_n(1);
  box p1 v box p2 v box p3 v box p5 v (dia (~p1 & box p1) v dia (~p1 & box p3) v false v (false v false) v (dia (~p3 & box p1) v
dia (~p3 & box p3) v false) v (false v false) v (dia (~p5 & box p1) v dia (~p5 & box p3) v false) v (false v false) v (false v false)
v (false v (dia (~p4 & box p2) v dia (~p6 & box p2))) v (false v false) v (false v (dia (~p4 & box p4) v dia (~p6 & box p4))) v (false
v false) v (false v (dia (~p4 & box p6) v dia (~p6 & box p6)))) v (dia dia ~p2 v dia dia ~p4 v dia dia ~p5 v dia dia ~p6)
> k_path_n(2);
  box p1 v box p2 v box p3 v box p5 v (dia (~p1 & box p1) v dia (~p1 & box p3) v false v (false v false) v (dia (~p3 & box p1) v
dia (~p3 & box p3) v false) v (false v false) v (dia (~p5 & box p1) v dia (~p5 & box p3) v false) v (false v false) v (dia dia (~p1
& box p1) v dia dia (~p1 & box p3) v dia dia (~p1 & box p5) v false) v (false v (dia (~p4 & box p2) v dia (~p6 & box p2))) v (dia dia
(~p3 & box p1) v dia dia (~p3 & box p3) v dia dia (~p3 & box p5) v false) v (false v (dia (~p4 & box p4) v dia (~p6 & box p4))) v (dia
dia (~p5 & box p1) v dia dia (~p5 & box p3) v dia dia (~p5 & box p2) v dia dia (~p5 & box p5) v false) v (false v (dia (~p4 & box p6)
v dia (~p6 & box p6))) v (false v false) v (false v (dia dia (~p2 & box p2) v dia dia (~p4 & box p2) v dia dia (~p6 & box p2))) v (false
v false) v (false v (dia dia (~p2 & box p4) v dia dia (~p4 & box p4) v dia dia (~p6 & box p4))) v (false v false) v (false v (dia dia
(~p2 & box p6) v dia dia (~p4 & box p6) v dia dia (~p6 & box p6)))) v (dia dia dia ~p2 v dia dia dia ~p4 v dia dia dia ~p2 v dia dia
dia ~p6)
```

12.1.13 k_ph_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_ph_p(i)), ", ", modaldepth(k_ph_p(i)), ", ", vars(k_ph_p(i)));
  1:  9, 1, [p101, p201]
  2:  40, 2, [p101, p102, p201, p202, p301, p302]
  3:  109, 2, [p101, p102, p103, p201, p202, p203, p301, p302, p303, p401, p402, p403]
  4:  231, 2, [p101, p102, p103, p104, p201, p202, p203, p204, p301, p302, p303, p304, p401, p402, p403, p404, p501, p502, p503,
p504]
  5:  421, 2, [p101, p102, p103, p104, p105, p201, p202, p203, p204, p205, p301, p302, p303, p304, p305, p401, p402, p403, p404,
p405, p501, p502, p503, p504, p505, p601, p602, p603, p604, p605]
  6:  694, 2, [p101, p102, p103, p104, p105, p106, p201, p202, p203, p204, p205, p206, p301, p302, p303, p304, p305, p306, p401,
p402, p403, p404, p405, p406, p501, p502, p503, p504, p505, p506, p601, p602, p603, p604, p605, p606, p701, p702, p703, p704, p705,
p706]
> k_ph_p(1);
  dia (p101 & p201) -> dia (p101 & p201)
> k_ph_p(2);
  dia ((p101 v box p102) & (p201 v p202) & (p301 v p302)) -> dia (p101 & p201 v p101 & p301 v p201 & p301 v box p102 & p202 v box
p102 & p302 v p202 & p302)
```

12.1.14 k_ph_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_ph_n(i)), ", ", modaldepth(k_ph_n(i)), ", ", vars(k_ph_n(i)));
  1:  10, 1, [p101, p201]
  2:  42, 2, [p101, p102, p201, p202, p301, p302]
  3:  112, 2, [p101, p102, p103, p201, p202, p203, p301, p302, p303, p401, p402, p403]
```

```
4:  235, 2, [p101, p102, p103, p104, p201, p202, p203, p204, p301, p302, p303, p304, p401, p402, p403, p404, p501, p502, p503, p504]
5:  426, 2, [p101, p102, p103, p104, p105, p201, p202, p203, p204, p205, p301, p302, p303, p304, p305, p401, p402, p403, p404, p405, p501, p502, p503, p504, p505, p601, p602, p603, p604, p605]
6:  700, 2, [p101, p102, p103, p104, p105, p106, p201, p202, p203, p204, p205, p206, p301, p302, p303, p304, p305, p306, p401, p402, p403, p404, p405, p406, p501, p502, p503, p504, p505, p506, p601, p602, p603, p604, p605, p606, p701, p702, p703, p704, p705, p706]
> k_ph_n(1);
dia (p101 & p201) -> dia (~p101 & p201)
> k_ph_n(2);
dia ((p101 v box p102) & (p201 v p202) & (p301 v p302)) -> dia (p101 & p201 v p101 & p301 v p201 & p301 v box p102 & ~p202 v box p102 & p302 v ~p202 & p302)
```

12.1.15 k_poly_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_poly_p(i)), ", ", modaldepth(k_poly_p(i)), ", ", vars(k_poly_p(i)));
1:  59, 4, [p1, p2, p3, p4, p5, p6, p8]
2:  141, 8, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p12, p14, p16]
3:  188, 10, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p14, p16, p18, p20]
4:  294, 14, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p18, p20, p22, p24, p26, p28]
5:  353, 16, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p20, p22, p24, p26, p28, p30, p32]
6:  483, 20, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p24, p26, p28, p30, p32, p34, p36, p38, p40]
> k_poly_p(1);
box box box box (p1 & p2 & p3 & p4) v (dia (dia (dia (false v dia (p1 <-> p2)) v box p3 v dia dia (p2 <-> p3)) v box p4 v dia dia dia (p3 <-> p1)) v box p5) v box box box box (~p2 & ~p4 & ~p6 & ~p8)
> k_poly_p(2);
box box box box box box box box (p1 & p2 & p3 & p4 & p5 & p6 & p7 & p8) v (dia (dia (dia (dia (dia (dia (dia (false v dia (p1 <-> p2)) v box p3 v dia dia (p2 <-> p3)) v box p4 v dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia (p4 <-> p5)) v box p6 v dia dia dia dia dia (p5 <-> p6)) v box p7 v dia dia dia dia dia dia (p6 <-> p7)) v box p8 v dia dia dia dia dia dia dia (p7 <-> p1)) v box p9) v box box box box box box box box (~p2 & ~p4 & ~p6 & ~p8 & ~p10 & ~p12 & ~p14 & ~p16)
```

12.1.16 k_poly_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_poly_n(i)), ", ", modaldepth(k_poly_n(i)), ", ", vars(k_poly_n(i)));
1:  78, 5, [p1, p2, p3, p4, p5, p6, p8, p10]
2:  119, 7, [p1, p2, p3, p4, p5, p6, p7, p8, p10, p12, p14]
3:  213, 11, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p14, p16, p18, p20, p22]
4:  266, 13, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p16, p18, p20, p22, p24, p26]
5:  384, 17, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p20, p22, p24, p26, p28, p30, p32, p34]
6:  449, 19, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p22, p24, p26, p28, p30, p32, p34, p36, p38]
> k_poly_n(1);
box box box box box (p1 & p2 & p3 & p4 & p5) v (dia (dia (dia (dia (false v dia (p1 <-> p2)) v box p3 v dia dia (p2 <-> p3)) v box p4 v dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia (p4 <-> p1)) v box p6) v box box box box box (~p2 & ~p4 & ~p6 & ~p8 & ~p10)
> k_poly_n(2);
box box box box box box box (p1 & p2 & p3 & p4 & p5 & p6 & p7) v (dia (dia (dia (dia (dia (dia (false v dia (p1 <-> p2)) v box p3 v dia dia (p2 <-> p3)) v box p4 v dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia (p4 <-> p5)) v box p6 v dia dia dia dia dia (p5 <-> p6)) v box p7 v dia dia dia dia dia dia (p6 <-> p1)) v box p8) v box box box box box box box (~p2 & ~p4 & ~p6 & ~p8 & ~p10 & ~p12 & ~p14)
```

12.1.17 k_t4p_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(k_t4p_p(i)), ", ", modaldepth(k_t4p_p(i)), ", ", vars(k_t4p_p(i)));
1:  110, 6, [p0, p1, p3, p4]
2:  153, 7, [p0, p1, p3, p4]
3:  196, 8, [p0, p1, p3, p4]
4:  239, 9, [p0, p1, p3, p4]
5:  282, 10, [p0, p1, p3, p4]
6:  325, 11, [p0, p1, p3, p4]
> k_t4p_p(1);
dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (box dia p0 & box (~p0 v box p3) v dia (box dia box (~p0 v box p3) & box dia p0) v box dia p0 & dia dia box (~p0 v box p3) v dia (dia box dia p0 & p0 & dia (~p0 v box p3)) v dia box (~p0 v box p3)) v dia (box dia p1 & dia dia box ~p1)) v dia p4
> k_t4p_p(2);
dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (dia (box p1 & dia dia ~p1) v dia (box dia p0 & box (~p0 v box p3) v dia (box dia box (~p0 v box p3) & box dia p0) v box dia p0 & dia dia box (~p0 v box p3) v dia (dia box dia p0 & p0 & dia (~p0 v box p3)) v dia box (~p0 v box p3)) v dia (box dia p1 & dia dia box ~p1)) v dia (box dia p1 & dia dia box ~p1)) v dia p4
```

12.1.18 k_t4p_n

```
> for i := 1 to 6 do
    print(i, ": ", length(k_t4p_n(i)), ", ", modaldepth(k_t4p_n(i)), ", ", vars(k_t4p_n(i)));
  1:  102, 6, [p0, p1, p3, p4]
  2:  188, 8, [p0, p1, p3, p4]
  3:  274, 10, [p0, p1, p3, p4]
  4:  360, 12, [p0, p1, p3, p4]
  5:  446, 14, [p0, p1, p3, p4]
  6:  532, 16, [p0, p1, p3, p4]
> k_t4p_n(1);
  dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (box dia p0 &
box (~p0 v box p3) v dia (box dia box (~p0 v box p3) & box dia p0) v box dia p0 & dia dia box (~p0 v box p3) v dia (dia box dia p0
& p0 & dia (~p0 v box p3))) v dia (box dia p1 & dia dia box ~p1)) v dia p4
> k_t4p_n(2);
  dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box
(~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (dia (box
p1 & dia dia ~p1) v dia (dia (box p1 & dia dia ~p1) v dia (box dia p0 & box (~p0 v box p3) v dia (box dia box (~p0 v box p3) & box
dia p0) v box dia p0 & dia dia box (~p0 v box p3) v dia (dia box dia p0 & p0 & dia (~p0 v box p3))) v dia (box dia p1 & dia dia box
~p1)) v dia (box dia p1 & dia dia box ~p1)) v dia (box dia p1 & dia dia box ~p1)) v dia p4
```

12.2 KT

12.2.1 kt_45_p

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_45_p(i)), ", ", modaldepth(kt_45_p(i)), ", ", vars(kt_45_p(i)));
  1:  41, 4, [p0]
  2:  89, 5, [p0]
  3:  143, 6, [p0]
  4:  203, 7, [p0]
  5:  269, 8, [p0]
  6:  341, 9, [p0]
> kt_45_p(1);
  box (dia ~p0 v box box p0) v dia box false v dia (dia box p0 & dia box dia ~p0) v dia dia (dia ~p0 & dia box p0) v dia (p0 & dia
box ~p0)
> kt_45_p(2);
  box box (dia ~p0 v box box p0) v dia box false v dia (dia box p0 & dia box dia ~p0) v dia dia (dia ~p0 & dia box p0) v dia (p0
& dia box ~p0) v (box box (dia ~p0 v box box p0) v dia dia box false v dia dia (dia box p0 & dia box dia ~p0) v dia dia dia (dia ~p0
& dia box p0) v dia dia (p0 & dia box ~p0))
```

12.2.2 kt_45_n

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_45_n(i)), ", ", modaldepth(kt_45_n(i)), ", ", vars(kt_45_n(i)));
  1:  67, 4, [p0]
  2:  142, 5, [p0]
  3:  224, 6, [p0]
  4:  313, 7, [p0]
  5:  409, 8, [p0]
  6:  512, 9, [p0]
> kt_45_n(1);
  box (box p0 v box dia ~p0) v dia (box p0 & dia dia ~p0) v dia (box dia p0 & dia dia box ~p0) v dia (box p0 & ~p0) v dia (box (box
~p0 v p0) & dia dia (dia p0 & ~p0)) v dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0))
> kt_45_n(2);
  box box (box p0 v box dia ~p0) v dia (box p0 & dia dia ~p0) v dia (box dia p0 & dia dia box ~p0) v dia (box p0 & ~p0) v dia (box
(box ~p0 v p0) & dia dia (dia p0 & ~p0)) v dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0)) v (box box (box p0 v box dia ~p0) v dia
dia (box p0 & dia dia ~p0) v dia dia (box dia p0 & dia dia box ~p0) v dia dia (box p0 & ~p0) v dia dia (box (box ~p0 v p0) & dia dia
(dia p0 & ~p0)) v dia dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0)))
```

12.2.3 kt_branch_p

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_branch_p(i)), ", ", modaldepth(kt_branch_p(i)), ", ", vars(kt_branch_p(i)));
  1:  76, 2, [p0, p1, p100, p101, p102]
  2:  122, 3, [p0, p1, p2, p100, p101, p102, p103]
  3:  168, 4, [p0, p1, p2, p3, p100, p101, p102, p103, p104]
  4:  214, 5, [p0, p1, p2, p3, p4, p100, p101, p102, p103, p104, p105]
  5:  260, 6, [p0, p1, p2, p3, p4, p5, p100, p101, p102, p103, p104, p105, p106]
  6:  306, 7, [p0, p1, p2, p3, p4, p5, p6, p100, p101, p102, p103, p104, p105, p106, p107]
> kt_branch_p(1);
```

```
    ~(p100 & ~p101 & box ((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101
-> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1))))
v ~box p1
> kt_branch_p(2);
    ~(p100 & ~p101 & box box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100
-> ~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102
-> ~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia
(p102 & ~p103 & ~p2))))) v ~box box p1
```

12.2.4 kt_branch_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_branch_n(i)), ", ", modaldepth(kt_branch_n(i)), ", ", vars(kt_branch_n(i)));
  1:  72, 2, [p0, p1, p100, p101, p102]
  2:  117, 3, [p0, p1, p2, p100, p101, p102, p103]
  3:  162, 4, [p0, p1, p2, p3, p100, p101, p102, p103, p104]
  4:  207, 5, [p0, p1, p2, p3, p4, p100, p101, p102, p103, p104, p105]
  5:  252, 6, [p0, p1, p2, p3, p4, p5, p100, p101, p102, p103, p104, p105, p106]
  6:  297, 7, [p0, p1, p2, p3, p4, p5, p6, p100, p101, p102, p103, p104, p105, p106, p107]
> kt_branch_n(1);
    ~(p100 & ~p101 & box ((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101
-> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1))))
> kt_branch_n(2);
    ~(p100 & ~p101 & box box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100
-> ~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102
-> ~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia
(p102 & ~p103 & ~p2)))))
```

12.2.5 kt_dum_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_dum_p(i)), ", ", modaldepth(kt_dum_p(i)), ", ", vars(kt_dum_p(i)));
  1:  95, 5, [p0]
  2:  103, 5, [p0]
  3:  103, 5, [p0]
  4:  114, 6, [p0]
  5:  124, 7, [p0]
  6:  137, 7, [p0]
> kt_dum_p(1);
    true & ~box (box (box (p0 -> box p0) -> p0) -> dia box p0 -> box p0) -> dia ~((box (box (p0 -> box p0) -> p0) -> box box (box (p0
-> box p0) -> p0)) & box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0) & (box (box ((p0 -> box p0)
-> box (p0 -> box p0)) -> p0 -> box p0) -> dia box (p0 -> box p0) -> p0 -> box p0)) v false
> kt_dum_p(2);
    box (box p0 -> box box p0) & ~box box (box (box (p0 -> box p0) -> p0) -> dia box p0 -> box p0) -> dia dia ~((box (box (p0 -> box
p0) -> p0) -> box box (box (p0 -> box p0) -> p0)) & box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0)
& (box (box ((p0 -> box p0) -> box (p0 -> box p0)) -> p0 -> box p0) -> dia box (p0 -> box p0) -> p0 -> box p0)) v false
```

12.2.6 kt_dum_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_dum_n(i)), ", ", modaldepth(kt_dum_n(i)), ", ", vars(kt_dum_n(i)));
  1:  90, 5, [p0]
  2:  114, 6, [p0]
  3:  114, 6, [p0]
  4:  141, 7, [p0]
  5:  167, 7, [p0]
  6:  196, 8, [p0]
> kt_dum_n(1);
    true & ~box (box (box (p0 -> box p0) -> p0) -> p0) -> dia ~((box (box (p0 -> box p0) -> p0) -> box box (box (p0 -> box p0) -> p0))
& box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0) & (box (box ((p0 -> box p0) -> box (p0 -> box p0))
-> p0 -> box p0) -> dia box (p0 -> box p0) -> p0 -> box p0)) v false
> kt_dum_n(2);
    box (box (box p0 -> box box p0) & (box (box (p0 -> box p0) -> p0) -> dia box p0 -> p0)) & ~box box (box (box (p0 -> box p0) ->
p0) -> p0) -> dia dia ~((box (box (p0 -> box p0) -> p0) -> box box (box (p0 -> box p0) -> p0)) & box (box p0 -> box box p0) & (box
(box (p0 -> box p0) -> p0) -> dia box p0 -> p0) & (box (box ((p0 -> box p0) -> box (p0 -> box p0)) -> p0 -> box p0) -> dia box (p0
-> box p0) -> p0 -> box p0)) v false
```

12.2.7 kt_grz_p

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_grz_p(i)), ", ", modaldepth(kt_grz_p(i)), ", ", vars(kt_grz_p(i)));
  1:  180, 7, [p0, p1, p2, p3]
  2:  201, 7, [p0, p1, p2, p3]
  3:  232, 7, [p0, p1, p2, p3]
  4:  271, 7, [p0, p1, p2, p3]
  5:  318, 7, [p0, p1, p2, p3, p4]
  6:  377, 7, [p0, p1, p2, p3, p4]
> kt_grz_p(1);
  box (box (box (p2 -> box p2) -> p2) -> p2) & true & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) ->
box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box
p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (p2 ->
box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (box (p1 -> box p1) -> p1) -> box
p1) v (box (box (p2 -> box p2) -> p2) -> box p2) & dia box ~p0 v (box (box (p3 -> box p3) -> p3) -> box p3) v (box (box (p2 -> box
p2) -> p2) -> box p2) & dia p0
> kt_grz_p(2);
  box (box (box (p2 -> box p2) -> p2) -> p2) & (box (box (box false v p1 -> box (box false v p1)) -> box false v p1) -> box false
v p1) & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box ((box
(p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box
(box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2)
-> box box (box (p2 -> box p2) -> p2))) -> (box (box (p1 -> box p1) -> p1) -> box p1) v (box (box (p2 -> box p2) -> p2) -> box p2)
& dia box ~p0 v (box (box (p3 -> box p3) -> p3) -> box p3) v (box (box (p2 -> box p2) -> p2) -> box p2) & dia p0
```

12.2.8 kt_grz_n

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_grz_n(i)), ", ", modaldepth(kt_grz_n(i)), ", ", vars(kt_grz_n(i)));
  1:  153, 7, [p1, p2, p3]
  2:  175, 7, [p1, p2, p3]
  3:  207, 7, [p1, p2, p3]
  4:  247, 7, [p1, p2, p3]
  5:  295, 7, [p1, p2, p3, p4]
  6:  355, 7, [p1, p2, p3, p4]
> kt_grz_n(1);
  box (box (box (p2 -> box p2) -> p2) -> box p2) & true & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2)
-> box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 ->
box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> box ((box
(p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (~box p1 -> box ~box p1) v (~box
p2 -> box ~box p2) v (~box p3 -> box ~box p3)
> kt_grz_n(2);
  box (box (box (p2 -> box p2) -> p2) -> box p2) & (box (box (box false v p1 -> box (box false v p1)) -> box false v p1) -> box (box
false v p1)) & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box
((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2)
& (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box
p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (~box p1 -> box ~box p1) v (~box p2 -> box ~box p2) v (~box p3 -> box ~box p3)
```

12.2.9 kt_md_p

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_md_p(i)), ", ", modaldepth(kt_md_p(i)), ", ", vars(kt_md_p(i)));
  1:  6, 0, [p1]
  2:  30, 3, [p1, p2]
  3:  73, 7, [p1, p2]
  4:  142, 13, [p1, p2]
  5:  246, 21, [p1, p2]
  6:  394, 31, [p1, p2]
> kt_md_p(1);
  p1 v false v ~p1
> kt_md_p(2);
  p1 v (~p1 & dia box dia p2 v ~p1 & dia box dia p1 v ~p2 & dia box dia p1) v dia box ~p1
```

12.2.10 kt_md_n

```
> for i := 1 to 6 do
    print(i, ": ", length(kt_md_n(i)), ", ", modaldepth(kt_md_n(i)), ", ", vars(kt_md_n(i)));
  1:  3, 0, [p1]
  2:  9, 3, [p1]
  3:  22, 7, [p1]
  4:  43, 13, [p1]
  5:  75, 21, [p1]
  6:  121, 31, [p1]
> kt_md_n(1);
  p1 v false
> kt_md_n(2);
  p1 v ~p1 & dia box dia p1
```

12.2.11 kt_path_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_path_p(i)), ", ", modaldepth(kt_path_p(i)), ", ", vars(kt_path_p(i)));
  1:  51, 1, [p11, p12, p13, p14, p15, p16]
  2:  156, 2, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26]
  3:  324, 3, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36]
  4:  455, 4, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46]
  5:  661, 5, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56]
  6:  818, 6, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56, p61, p62, p63, p64, p65, p66]
> kt_path_p(1);
  box p11 v box p12 v box p13 v box p15 v (false v false v (false v false) v (false v false) v (false v false) v (false v false)
v (false v false)) v (dia ~p12 v dia ~p14 v dia ~p12 v dia ~p16)
> kt_path_p(2);
  box p11 v box p12 v box p13 v box p15 v (dia (~p11 & box p21) v dia (~p11 & box p23) v false v (dia (~p12 & box p25) v false) v
(dia (~p13 & box p21) v dia (~p13 & box p23) v false) v (false v false) v (dia (~p15 & box p21) v dia (~p15 & box p23) v false) v (false
v false) v (false v false) v (false v (dia (~p14 & box p22) v dia (~p16 & box p22))) v (false v false) v (false v (dia (~p14 & box
p24) v dia (~p16 & box p24))) v (false v false) v (false v (dia (~p14 & box p26) v dia (~p16 & box p26)))) v (dia dia ~p22 v dia dia
~p24 v dia dia ~p25 v dia dia ~p26)
```

12.2.12 kt_path_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_path_n(i)), ", ", modaldepth(kt_path_n(i)), ", ", vars(kt_path_n(i)));
  1:  151, 2, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26]
  2:  319, 3, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36]
  3:  447, 4, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46]
  4:  653, 5, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56]
  5:  811, 6, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56, p61, p62, p63, p64, p65, p66]
  6:  1055, 7, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56, p61, p62, p63, p64, p65, p66, p71, p72, p73, p74, p75, p76]
> kt_path_n(1);
  box p11 v box p12 v box p13 v box p15 v (dia (~p11 & box p21) v dia (~p11 & box p23) v false v (false v false) v (dia (~p13 & box
p21) v dia (~p13 & box p23) v false) v (false v false) v (dia (~p15 & box p21) v dia (~p15 & box p23) v false) v (false v false) v
(false v false) v (false v (dia (~p14 & box p22) v dia (~p16 & box p22))) v (false v false) v (false v (dia (~p14 & box p24) v dia
(~p16 & box p24))) v (false v false) v (false v (dia (~p14 & box p26) v dia (~p16 & box p26)))) v (dia dia ~p22 v dia dia ~p24 v dia
dia ~p25 v dia dia ~p26)
> kt_path_n(2);
  box p11 v box p12 v box p13 v box p15 v (dia (~p11 & box p21) v dia (~p11 & box p23) v false v (false v false) v (dia (~p13 & box
p21) v dia (~p13 & box p23) v false) v (false v false) v (dia (~p15 & box p21) v dia (~p15 & box p23) v false) v (false v false) v
(dia dia (~p21 & box p31) v dia dia (~p21 & box p33) v dia dia (~p21 & box p35) v false) v (false v (dia (~p14 & box p22) v dia (~p16
& box p22))) v (dia dia (~p23 & box p31) v dia dia (~p23 & box p33) v dia dia (~p23 & box p35) v false) v (false v (dia (~p14 & box
p24) v dia (~p16 & box p24))) v (dia dia (~p25 & box p31) v dia dia (~p25 & box p33) v dia dia (~p25 & box p32) v dia dia (~p25 & box
p35) v false) v (false v (dia (~p14 & box p26) v dia (~p16 & box p26))) v (false v false) v (false v (dia dia (~p22 & box p32) v dia
dia (~p24 & box p32) v dia dia (~p26 & box p32))) v (false v false) v (false v (dia dia (~p22 & box p34) v dia dia (~p24 & box p34)
v dia dia (~p26 & box p34))) v (false v false) v (false v (dia dia (~p22 & box p36) v dia dia (~p24 & box p36) v dia dia (~p26 & box
p36)))) v (dia dia dia ~p32 v dia dia dia ~p34 v dia dia dia ~p32 v dia dia dia ~p36)
```

12.2.13 kt_ph_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_ph_p(i)), ", ", modaldepth(kt_ph_p(i)), ", ", vars(kt_ph_p(i)));
  1:  8, 1, [p101, p201]
  2:  39, 2, [p101, p102, p201, p202, p301, p302]
  3:  108, 2, [p101, p102, p103, p201, p202, p203, p301, p302, p303, p401, p402, p403]
  4:  230, 2, [p101, p102, p103, p104, p201, p202, p203, p204, p301, p302, p303, p304, p401, p402, p403, p404, p501, p502, p503,
p504]
  5:  420, 2, [p101, p102, p103, p104, p105, p201, p202, p203, p204, p205, p301, p302, p303, p304, p305, p401, p402, p403, p404,
p405, p501, p502, p503, p504, p505, p601, p602, p603, p604, p605]
  6:  693, 2, [p101, p102, p103, p104, p105, p106, p201, p202, p203, p204, p205, p206, p301, p302, p303, p304, p305, p306, p401,
p402, p403, p404, p405, p406, p501, p502, p503, p504, p505, p506, p601, p602, p603, p604, p605, p606, p701, p702, p703, p704, p705,
p706]
> kt_ph_p(1);
  p101 & p201 -> dia (p101 & p201)
> kt_ph_p(2);
  (p101 v box p102) & (p201 v p202) & (p301 v p302) -> dia (p101 & p201 v p101 & p301 v p201 & p301 v box p102 & p202 v box p102
& p302 v p202 & p302)
```

12.2.14 kt_ph_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_ph_n(i)), ", ", modaldepth(kt_ph_n(i)), ", ", vars(kt_ph_n(i)));
  1:  9, 1, [p101, p201]
  2:  41, 2, [p101, p102, p201, p202, p301, p302]
  3:  111, 2, [p101, p102, p103, p201, p202, p203, p301, p302, p303, p401, p402, p403]
  4:  234, 2, [p101, p102, p103, p104, p201, p202, p203, p204, p301, p302, p303, p304, p401, p402, p403, p404, p501, p502, p503,
p504]
  5:  425, 2, [p101, p102, p103, p104, p105, p201, p202, p203, p204, p205, p301, p302, p303, p304, p305, p401, p402, p403, p404,
p405, p501, p502, p503, p504, p505, p601, p602, p603, p604, p605]
  6:  699, 2, [p101, p102, p103, p104, p105, p106, p201, p202, p203, p204, p205, p206, p301, p302, p303, p304, p305, p306, p401,
p402, p403, p404, p405, p406, p501, p502, p503, p504, p505, p506, p601, p602, p603, p604, p605, p606, p701, p702, p703, p704, p705,
p706]
> kt_ph_n(1);
  p101 & p201 -> dia (~p101 & p201)
> kt_ph_n(2);
  (p101 v box p102) & (p201 v p202) & (p301 v p302) -> dia (p101 & p201 v p101 & p301 v p201 & p301 v box p102 & ~p202 v box p102
& p302 v ~p202 & p302)
```

12.2.15 kt_poly_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_poly_p(i)), ", ", modaldepth(kt_poly_p(i)), ", ", vars(kt_poly_p(i)));
  1:  108, 8, [p1, p2, p3, p4, p5, p6, p7, p8, p10, p12]
  2:  261, 14, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p16, p18, p20, p22, p24]
  3:  383, 18, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p20, p22, p24, p26, p28, p30, p32]
  4:  596, 24, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p26,
p28, p30, p32, p34, p36, p38, p40, p42, p44]
  5:  758, 28, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25,
p26, p27, p28, p30, p32, p34, p36, p38, p40, p42, p44, p46, p48, p50, p52]
  6:  1031, 34, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25,
p26, p27, p28, p29, p30, p31, p32, p33, p34, p36, p38, p40, p42, p44, p46, p48, p50, p52, p54, p56, p58, p60, p62, p64]
> kt_poly_p(1);
  box box box box box box (p1 & p2 & p3 & p4 & p5 & p6) v (dia (dia (dia (dia (dia (false v dia dia dia (p1 <-> p2)) v box p3 v dia
dia dia dia (p2 <-> p3)) v box p4 v dia dia dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia dia dia (p4 <-> p5)) v box p6 v dia
dia dia dia dia dia dia (p5 <-> p1)) v box p7) v box box box box box box (~p2 & ~p4 & ~p6 & ~p8 & ~p10 & ~p12)
> kt_poly_p(2);
  box box box box box box box box box box box box (p1 & p2 & p3 & p4 & p5 & p6 & p7 & p8 & p9 & p10 & p11 & p12) v (dia (dia (dia
(dia (dia (dia (dia (dia (dia (dia (false v dia dia dia (p1 <-> p2)) v box p3 v dia dia dia dia (p2 <-> p3)) v box p4 v dia dia
dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia dia dia (p4 <-> p5)) v box p6 v dia dia dia dia dia dia dia (p5 <-> p6)) v box
p7 v dia dia dia dia dia dia dia dia (p6 <-> p7)) v box p8 v dia dia dia dia dia dia dia dia dia (p7 <-> p8)) v box p9 v dia dia dia
dia dia dia dia dia dia dia (p8 <-> p9)) v box p10 v dia dia dia dia dia dia dia dia dia dia dia (p9 <-> p10)) v box p11 v dia dia
dia dia dia dia dia dia dia dia dia (p10 <-> p11)) v box p12 v dia dia dia dia dia dia dia dia dia dia dia dia dia (p11 <-> p1))
v box p13) v box box box box box box box box box box box box (~p2 & ~p4 & ~p6 & ~p8 & ~p10 & ~p12 & ~p14 & ~p16 & ~p18 & ~p20 & ~p22
& ~p24)
```

12.2.16 kt_poly_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_poly_n(i)), ", ", modaldepth(kt_poly_n(i)), ", ", vars(kt_poly_n(i)));
  1:  86, 7, [p1, p2, p3, p4, p5, p6, p8, p10]
  2:  131, 9, [p1, p2, p3, p4, p5, p6, p7, p8, p10, p12, p14]
  3:  233, 13, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p14, p16, p18, p20, p22]
  4:  290, 15, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p16, p18, p20, p22, p24, p26]
  5:  416, 19, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p20, p22, p24, p26, p28, p30, p32,
p34]
  6:  485, 21, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p22, p24, p26, p28, p30,
p32, p34, p36, p38]
> kt_poly_n(1);
  box box box box box (p1 & p2 & p3 & p4 & p5) v (dia (dia (dia (dia (false v dia dia dia (p1 <-> p2)) v box p3 v dia dia dia dia
(p2 <-> p3)) v box p4 v dia dia dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia dia dia (p4 <-> p1)) v box p6) v box box box box
box (~p2 & ~p4 & ~p6 & ~p8 & ~p10)
> kt_poly_n(2);
  box box box box box box box (p1 & p2 & p3 & p4 & p5 & p6 & p7) v (dia (dia (dia (dia (dia (dia (false v dia dia dia (p1 <-> p2))
v box p3 v dia dia dia dia (p2 <-> p3)) v box p4 v dia dia dia dia dia (p3 <-> p4)) v box p5 v dia dia dia dia dia dia (p4 <-> p5))
v box p6 v dia dia dia dia dia dia dia (p5 <-> p6)) v box p7 v dia dia dia dia dia dia dia dia (p6 <-> p1)) v box p8) v box box box
box box box box (~p2 & ~p4 & ~p6 & ~p8 & ~p10 & ~p12 & ~p14)
```

12.2.17 kt_t4p_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_t4p_p(i)), ", ", modaldepth(kt_t4p_p(i)), ", ", vars(kt_t4p_p(i)));
  1:  74, 5, [p0, p1, p3, p4]
  2:  106, 6, [p0, p1, p3, p4]
  3:  138, 7, [p0, p1, p3, p4]
  4:  170, 8, [p0, p1, p3, p4]
  5:  202, 9, [p0, p1, p3, p4]
  6:  234, 10, [p0, p1, p3, p4]
> kt_t4p_p(1);
  dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (box dia p0 &
dia dia box (~p0 v box p3) v dia (dia box dia p0 & p0 & dia (~p0 v box p3)) v dia box (~p0 v box p3))) v dia p4
> kt_t4p_p(2);
  dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1)) v box (~box
p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (dia (box p1 & dia dia ~p1) v dia (box dia p0 & dia dia box (~p0 v box p3) v
dia (dia box dia p0 & p0 & dia (~p0 v box p3)) v dia box (~p0 v box p3)))) v dia p4
```

12.2.18 kt_t4p_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(kt_t4p_n(i)), ", ", modaldepth(kt_t4p_n(i)), ", ", vars(kt_t4p_n(i)));
  1:  77, 5, [p0, p1, p3, p4]
  2:  141, 7, [p0, p1, p3, p4]
  3:  205, 9, [p0, p1, p3, p4]
  4:  269, 11, [p0, p1, p3, p4]
  5:  333, 13, [p0, p1, p3, p4]
  6:  397, 15, [p0, p1, p3, p4]
> kt_t4p_n(1);
  dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (box dia p0 &
box (~p0 v box p3) v box dia p0 & dia dia box (~p0 v box p3) v dia (dia box dia p0 & p0 & dia (~p0 v box p3)))) v dia p4
> kt_t4p_n(2);
  dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box
(~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1) v (dia (box p1 & dia dia ~p1) v dia (dia (box
p1 & dia dia ~p1) v dia (dia (box p1 & dia dia ~p1) v dia (box dia p0 & box (~p0 v box p3) v box dia p0 & dia dia box (~p0 v box p3)
v dia (dia box dia p0 & p0 & dia (~p0 v box p3)))))) v dia p4
```

12.3 S4

12.3.1 s4_45_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_45_p(i)), ", ", modaldepth(s4_45_p(i)), ", ", vars(s4_45_p(i)));
  1:  79, 4, [p0, p1]
  2:  209, 5, [p0, p1]
  3:  389, 6, [p0, p1]
  4:  619, 7, [p0, p1]
  5:  899, 8, [p0, p1]
  6:  1229, 9, [p0, p1]
> s4_45_p(1);
  box p0 v box (box p0 v box dia ~p0) v box p1 v dia ~p0 & dia box false & dia ~p1 v dia ~p0 & dia (dia box p0 & dia box dia ~p0)
& dia ~p1 v dia ~p0 & dia dia (dia ~p0 & dia box p0) & dia ~p1 v dia ~p0 & dia (p0 & dia box ~p0) & dia ~p1
> s4_45_p(2);
  box p0 v box (box p0 v box (box p0 v box dia ~p0) v box p1) v box p1 v dia ~p0 & dia box false & dia ~p1 v dia ~p0 & dia (dia box
p0 & dia box dia ~p0) & dia ~p1 v dia ~p0 & dia dia (dia ~p0 & dia box p0) & dia ~p1 v dia ~p0 & dia (p0 & dia box ~p0) & dia ~p1 v
(box p0 v box (box p0 v box (box p0 v box dia ~p0) v box p1) v box p1 v dia ~p0 & dia (dia ~p0 & dia box false & dia ~p1) & dia ~p1
v dia ~p0 & dia (dia ~p0 & dia (dia box p0 & dia box dia ~p0) & dia ~p1) & dia ~p1 v dia ~p0 & dia (dia ~p0 & dia dia (dia ~p0 & dia
box p0) & dia ~p1) & dia ~p1 v dia ~p0 & dia (dia ~p0 & dia (p0 & dia box ~p0) & dia ~p1) & dia ~p1)
```

12.3.2 s4_45_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_45_n(i)), ", ", modaldepth(s4_45_n(i)), ", ", vars(s4_45_n(i)));
  1:  111, 4, [p0, p1]
  2:  282, 5, [p0, p1]
  3:  512, 6, [p0, p1]
  4:  801, 7, [p0, p1]
  5:  1149, 8, [p0, p1]
  6:  1556, 9, [p0, p1]
> s4_45_n(1);
```

```
box p0 v box (box p0 v box dia ~p0) v box p1 v dia ~p0 & dia (box p0 & dia dia ~p0) & dia ~p1 v dia ~p0 & dia (box dia p0 & dia
dia box ~p0) & dia ~p1 v dia ~p0 & dia (box p0 & ~p0) & dia ~p1 v dia ~p0 & dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0)) & dia
~p1 v dia ~p0 & dia (box (dia ~p0 v p0) & (box p0 & ~p0)) & dia ~p1
> s4_45_n(2);
box p0 v box (box p0 v box (box p0 v box dia ~p0) v box p1) v box p1 v dia ~p0 & dia (box p0 & dia dia ~p0) & dia ~p1 v dia ~p0
& dia (box dia p0 & dia dia box ~p0) & dia ~p1 v dia ~p0 & dia (box p0 & ~p0) & dia ~p1 v dia ~p0 & dia (box (dia ~p0 v p0) & dia dia
(box p0 & ~p0)) & dia ~p1 v dia ~p0 & dia (box (dia ~p0 v p0) & (box p0 & ~p0)) & dia ~p1 v (box p0 v box (box p0 v box (box p0 v box
dia ~p0) v box p1) v box p1 v dia ~p0 & dia (dia ~p0 & dia (box p0 & dia dia ~p0) & dia ~p1) & dia ~p1 v dia ~p0 & dia (dia ~p0 & dia
(box dia p0 & dia dia box ~p0) & dia ~p1) & dia ~p1 v dia ~p0 & dia (dia ~p0 & dia (box p0 & ~p0) & dia ~p1) & dia ~p1 v dia ~p0 &
dia (dia ~p0 & dia (box (dia ~p0 v p0) & dia dia (box p0 & ~p0)) & dia ~p1) & dia ~p1 v dia ~p0 & dia (dia ~p0 & dia (box (dia ~p0
v p0) & (box p0 & ~p0)) & dia ~p1) & dia ~p1)
```

12.3.3 s4_branch_p

```
> for i := 1 to 6 do
    print(i, ": ", length(s4_branch_p(i)), ", ", modaldepth(s4_branch_p(i)), ", ", vars(s4_branch_p(i)));
1: 76, 2, [p0, p1, p100, p101, p102]
2: 120, 2, [p0, p1, p2, p100, p101, p102, p103]
3: 164, 2, [p0, p1, p2, p3, p100, p101, p102, p103, p104]
4: 208, 2, [p0, p1, p2, p3, p4, p100, p101, p102, p103, p104, p105]
5: 252, 2, [p0, p1, p2, p3, p4, p5, p100, p101, p102, p103, p104, p105, p106]
6: 296, 2, [p0, p1, p2, p3, p4, p5, p6, p100, p101, p102, p103, p104, p105, p106, p107]
> s4_branch_p(1);
~(p100 & ~p101 & box ((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101
-> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1))))
v ~box p1
> s4_branch_p(2);
~(p100 & ~p101 & box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100
-> ~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102
-> ~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia
(p102 & ~p103 & ~p2))))) v ~box p1
```

12.3.4 s4_branch_n

```
> for i := 1 to 6 do
    print(i, ": ", length(s4_branch_n(i)), ", ", modaldepth(s4_branch_n(i)), ", ", vars(s4_branch_n(i)));
1: 72, 2, [p0, p1, p100, p101, p102]
2: 116, 2, [p0, p1, p2, p100, p101, p102, p103]
3: 160, 2, [p0, p1, p2, p3, p100, p101, p102, p103, p104]
4: 204, 2, [p0, p1, p2, p3, p4, p100, p101, p102, p103, p104, p105]
5: 248, 2, [p0, p1, p2, p3, p4, p5, p100, p101, p102, p103, p104, p105, p106]
6: 292, 2, [p0, p1, p2, p3, p4, p5, p6, p100, p101, p102, p103, p104, p105, p106, p107]
> s4_branch_n(1);
~(p100 & ~p101 & box ((p101 -> p100) & (p102 -> p101) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100 -> ~p0))) & (p101
-> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1)))) & (p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1))))
> s4_branch_n(2);
~(p100 & ~p101 & box ((p101 -> p100) & (p102 -> p101) & (p103 -> p102) & ((p100 -> (p0 -> box (p100 -> p0)) & (~p0 -> box (p100
-> ~p0))) & (p101 -> (p1 -> box (p101 -> p1)) & (~p1 -> box (p101 -> ~p1))) & (p102 -> (p2 -> box (p102 -> p2)) & (~p2 -> box (p102
-> ~p2)))) & ((p100 & ~p101 -> dia (p101 & ~p102 & p1) & dia (p101 & ~p102 & ~p1)) & (p101 & ~p102 -> dia (p102 & ~p103 & p2) & dia
(p102 & ~p103 & ~p2)))))
```

12.3.5 s4_grz_p

```
> for i := 1 to 6 do
    print(i, ": ", length(s4_grz_p(i)), ", ", modaldepth(s4_grz_p(i)), ", ", vars(s4_grz_p(i)));
1: 186, 7, [p0, p1, p2, p3]
2: 207, 7, [p0, p1, p2, p3]
3: 238, 7, [p0, p1, p2, p3]
4: 277, 7, [p0, p1, p2, p3]
5: 324, 7, [p0, p1, p2, p3, p4]
6: 383, 7, [p0, p1, p2, p3, p4]
> s4_grz_p(1);
box (box (box (p2 -> box p2) -> p2) -> p2) & true & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) ->
box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box
p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (p2 ->
box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (box (p1 -> box p1) -> p1) -> box
p1) v (box (box (p2 -> box p2) -> p2) -> box p2) & ~box dia p0 v (box (box (p3 -> box p3) -> p3) -> box p3) v (box (box (p2 -> box
p2) -> p2) -> box p2) & ~dia ~(box dia p0 v p1)
> s4_grz_p(2);
box (box (box (p2 -> box p2) -> p2) -> p2) & (box (box (box false v p1 -> box (box false v p1)) -> box false v p1) -> box false
v p1) & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box ((box
(p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box
(box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2)
-> box box (box (p2 -> box p2) -> p2))) -> (box (box (p1 -> box p1) -> p1) -> box p1) v (box (box (p2 -> box p2) -> p2) -> box p2)
& ~box dia p0 v (box (box (p3 -> box p3) -> p3) -> box p3) v (box (box (p2 -> box p2) -> p2) -> box p2) & ~dia ~(box dia p0 v p1)
```

12.3.6 s4_grz_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_grz_n(i)), ", ", modaldepth(s4_grz_n(i)), ", ", vars(s4_grz_n(i)));
  1:  153, 7, [p1, p2, p3]
  2:  175, 7, [p1, p2, p3]
  3:  207, 7, [p1, p2, p3]
  4:  247, 7, [p1, p2, p3]
  5:  295, 7, [p1, p2, p3, p4]
  6:  355, 7, [p1, p2, p3, p4]
> s4_grz_n(1);
  box (box (box (p2 -> box p2) -> p2) -> box p2) & true & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2)
-> box box (box (p2 -> box p2) -> p2)) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 ->
box p2) -> p2)))) -> (box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> box ((box
(p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (~box p1 -> box ~box p1) v (~box
p2 -> box ~box p2) v (~box p3 -> box ~box p3)
> s4_grz_n(2);
  box (box (box (p2 -> box p2) -> p2) -> box p2) & (box (box (box false v p1 -> box (box false v p1)) -> box false v p1) -> box (box
false v p1)) & (box (box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)) -> box
((box (p2 -> box p2) -> p2) & (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (box (p2 -> box p2) -> p2)
& (box (box (p2 -> box p2) -> p2) -> box box (box (p2 -> box p2) -> p2))) -> box ((box (p2 -> box p2) -> p2) & (box (box (p2 -> box
p2) -> p2) -> box box (box (p2 -> box p2) -> p2)))) -> (~box p1 -> box ~box p1) v (~box p2 -> box ~box p2) v (~box p3 -> box ~box p3)
```

12.3.7 s4_ipc_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_ipc_p(i)), ", ", modaldepth(s4_ipc_p(i)), ", ", vars(s4_ipc_p(i)));
  1:  11, 3, [p1]
  2:  27, 3, [p1, p2]
  3:  49, 3, [p1, p2, p3]
  4:  77, 3, [p1, p2, p3, p4]
  5:  111, 3, [p1, p2, p3, p4, p5]
  6:  151, 3, [p1, p2, p3, p4, p5, p6]
> s4_ipc_p(1);
  box (box (box p1 -> box p1) -> false) -> false
> s4_ipc_p(2);
  box (box (box p1 -> box p1 & box p2) -> false) & box (box (box p2 -> box p1 & box p2) -> false) -> false
```

12.3.8 s4_ipc_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_ipc_n(i)), ", ", modaldepth(s4_ipc_n(i)), ", ", vars(s4_ipc_n(i)));
  1:  3, 0, []
  2:  16, 3, [p1, p2]
  3:  35, 3, [p1, p2, p3]
  4:  60, 3, [p1, p2, p3, p4]
  5:  91, 3, [p1, p2, p3, p4, p5]
  6:  128, 3, [p1, p2, p3, p4, p5, p6]
> s4_ipc_n(1);
  true -> false
> s4_ipc_n(2);
  true & box (box (box p2 -> box p1 & box p2) -> false) -> false
```

12.3.9 s4_md_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_md_p(i)), ", ", modaldepth(s4_md_p(i)), ", ", vars(s4_md_p(i)));
  1:  6, 0, [p1]
  2:  38, 3, [p1, p2]
  3:  117, 7, [p1, p2]
  4:  268, 13, [p1, p2]
  5:  518, 21, [p1, p2]
  6:  894, 31, [p1, p2]
> s4_md_p(1);
  p1 v false v ~p1
> s4_md_p(2);
  p1 v (~p1 & dia box (dia p2 v p2) v ~p1 & dia box (dia p1 v p2) v ~p2 & dia box (dia p1 v p2)) v (dia box ~p1 v p2)
```

12.3.10 s4_md_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_md_n(i)), ", ", modaldepth(s4_md_n(i)), ", ", vars(s4_md_n(i)));
  1:  3, 0, [p1]
  2:  11, 3, [p1, p2]
  3:  34, 7, [p1, p2]
  4:  79, 13, [p1, p2]
  5:  155, 21, [p1, p2]
  6:  271, 31, [p1, p2]
> s4_md_n(1);
  p1 v false
> s4_md_n(2);
  p1 v ~p1 & dia box (dia p1 v p2)
```

12.3.11 s4_path_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_path_p(i)), ", ", modaldepth(s4_path_p(i)), ", ", vars(s4_path_p(i)));
  1:  56, 2, [p11, p12, p13, p14, p15, p16]
  2:  157, 2, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26]
  3:  321, 3, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36]
  4:  448, 4, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46]
  5:  650, 5, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56]
  6:  803, 6, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56, p61, p62, p63, p64, p65, p66]
> s4_path_p(1);
  box box p11 v box box p12 v box box p13 v box box p15 v (false v false v (false v false) v (false v false) v (false v false) v
(false v false) v (false v false)) v dia (dia ~p12 v dia ~p14 v dia ~p12 v dia ~p16)
> s4_path_p(2);
  box box p11 v box box p12 v box box p13 v box box p15 v (dia (~p11 & box p21) v dia (~p11 & box p23) v false v (dia (~p12 & box
p25) v false) v (dia (~p13 & box p21) v dia (~p13 & box p23) v false) v (false v false) v (dia (~p15 & box p21) v dia (~p15 & box p23)
v false) v (false v false) v (false v false) v (false v (dia (~p14 & box p22) v dia (~p16 & box p22))) v (false v false) v (false v
(dia (~p14 & box p24) v dia (~p16 & box p24))) v (false v false) v (false v (dia (~p14 & box p26) v dia (~p16 & box p26)))) v dia (dia
~p22 v dia ~p24 v dia ~p25 v dia ~p26)
```

12.3.12 s4_path_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_path_n(i)), ", ", modaldepth(s4_path_n(i)), ", ", vars(s4_path_n(i)));
  1:  152, 2, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26]
  2:  316, 3, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36]
  3:  440, 4, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46]
  4:  642, 5, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56]
  5:  796, 6, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56, p61, p62, p63, p64, p65, p66]
  6:  1036, 7, [p11, p12, p13, p14, p15, p16, p21, p22, p23, p24, p25, p26, p31, p32, p33, p34, p35, p36, p41, p42, p43, p44, p45,
p46, p51, p52, p53, p54, p55, p56, p61, p62, p63, p64, p65, p66, p71, p72, p73, p74, p75, p76]
> s4_path_n(1);
  box box p11 v box box p12 v box box p13 v box box p15 v (dia (~p11 & box p21) v dia (~p11 & box p23) v false v (false v false)
v (dia (~p13 & box p21) v dia (~p13 & box p23) v false) v (false v false) v (dia (~p15 & box p21) v dia (~p15 & box p23) v false) v
(false v false) v (false v false) v (false v (dia (~p14 & box p22) v dia (~p16 & box p22))) v (false v false) v (false v (dia (~p14
& box p24) v dia (~p16 & box p24))) v (false v false) v (false v (dia (~p14 & box p26) v dia (~p16 & box p26)))) v dia (dia ~p22 v
dia ~p24 v dia ~p25 v dia ~p26)
> s4_path_n(2);
  box box p11 v box box p12 v box box p13 v box box p15 v (dia (~p11 & box p21) v dia (~p11 & box p23) v false v (false v false)
v (dia (~p13 & box p21) v dia (~p13 & box p23) v false) v (false v false) v (dia (~p15 & box p21) v dia (~p15 & box p23) v false) v
(false v false) v (dia dia (~p21 & box p31) v dia dia (~p21 & box p33) v dia dia (~p21 & box p35) v false) v (false v (dia (~p14 &
box p22) v dia (~p16 & box p22))) v (dia dia (~p23 & box p31) v dia dia (~p23 & box p33) v dia dia (~p23 & box p35) v false) v (false
v (dia (~p14 & box p24) v dia (~p16 & box p24))) v (dia dia (~p25 & box p31) v dia dia (~p25 & box p33) v dia dia (~p25 & box p32)
v dia dia (~p25 & box p35) v false) v (false v (dia (~p14 & box p26) v dia (~p16 & box p26))) v (false v false) v (false v (dia dia
(~p22 & box p32) v dia dia (~p24 & box p32) v dia dia (~p26 & box p32))) v (false v false) v (false v (dia dia (~p22 & box p34) v dia
dia (~p24 & box p34) v dia dia (~p26 & box p34))) v (false v false) v (false v (dia dia (~p22 & box p36) v dia dia (~p24 & box p36)
v dia dia (~p26 & box p36)))) v dia (dia ~p32 v dia ~p34 v dia ~p32 v dia ~p36)
```

12.3.13 s4_ph_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_ph_p(i)), ", ", modaldepth(s4_ph_p(i)), ", ", vars(s4_ph_p(i)));
  1:  9, 2, [p101, p201]
  2:  45, 3, [p101, p102, p201, p202, p301, p302]
  3:  126, 3, [p101, p102, p103, p201, p202, p203, p301, p302, p303, p401, p402, p403]
  4:  270, 3, [p101, p102, p103, p104, p201, p202, p203, p204, p301, p302, p303, p304, p401, p402, p403, p404, p501, p502, p503,
p504]
  5:  495, 3, [p101, p102, p103, p104, p105, p201, p202, p203, p204, p205, p301, p302, p303, p304, p305, p401, p402, p403, p404,
p405, p501, p502, p503, p504, p505, p601, p602, p603, p604, p605]
  6:  819, 3, [p101, p102, p103, p104, p105, p106, p201, p202, p203, p204, p205, p206, p301, p302, p303, p304, p305, p306, p401,
p402, p403, p404, p405, p406, p501, p502, p503, p504, p505, p506, p601, p602, p603, p604, p605, p606, p701, p702, p703, p704, p705,
p706]
> s4_ph_p(1);
  p101 & p201 -> dia dia (p101 & p201)
> s4_ph_p(2);
  (p101 v box p102) & (p201 v p202) & (p301 v p302) -> dia (dia (p101 & p201) v dia (p101 & p301) v dia (p201 & p301) v dia (box
p102 & p202) v dia (box p102 & p302) v dia (p202 & p302))
```

12.3.14 s4_ph_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_ph_n(i)), ", ", modaldepth(s4_ph_n(i)), ", ", vars(s4_ph_n(i)));
  1:  10, 2, [p101, p201]
  2:  47, 3, [p101, p102, p201, p202, p301, p302]
  3:  129, 3, [p101, p102, p103, p201, p202, p203, p301, p302, p303, p401, p402, p403]
  4:  274, 3, [p101, p102, p103, p104, p201, p202, p203, p204, p301, p302, p303, p304, p401, p402, p403, p404, p501, p502, p503,
p504]
  5:  500, 3, [p101, p102, p103, p104, p105, p201, p202, p203, p204, p205, p301, p302, p303, p304, p305, p401, p402, p403, p404,
p405, p501, p502, p503, p504, p505, p601, p602, p603, p604, p605]
  6:  825, 3, [p101, p102, p103, p104, p105, p106, p201, p202, p203, p204, p205, p206, p301, p302, p303, p304, p305, p306, p401,
p402, p403, p404, p405, p406, p501, p502, p503, p504, p505, p506, p601, p602, p603, p604, p605, p606, p701, p702, p703, p704, p705,
p706]
> s4_ph_n(1);
  p101 & p201 -> dia dia (~p101 & p201)
> s4_ph_n(2);
  (p101 v box p102) & (p201 v p202) & (p301 v p302) -> dia (dia (p101 & p201) v dia (p101 & p301) v dia (p201 & p301) v dia (box
p102 & ~p202) v dia (box p102 & p302) v dia (~p202 & p302))
```

12.3.15 s4_s5_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_s5_p(i)), ", ", modaldepth(s4_s5_p(i)), ", ", vars(s4_s5_p(i)));
  1:  30, 3, [p1, p2, p3]
  2:  78, 6, [p1, p2, p3, p4, p5, p6]
  3:  126, 9, [p1, p2, p3, p4, p5, p6, p7, p8, p9]
  4:  174, 12, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12]
  5:  222, 15, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15]
  6:  270, 18, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18]
> s4_s5_p(1);
  box dia (box (p1 <-> p2) v box p3 v (dia (p1 & ~p2 v ~p1 & p2) v box false)) v box (dia p1 -> ~p3)
> s4_s5_p(2);
  box dia (box ((p1 <-> p2) v (p2 <-> p3) v (p3 <-> p4) v (p4 <-> p5)) v box p6 v (dia (p1 & ~p2 v ~p1 & p2) v box (dia (p2 & ~p3
v ~p2 & p3) v box (dia (p3 & ~p4 v ~p3 & p4) v box (dia (p4 & ~p5 v ~p4 & p5) v box false))))) v box (dia p1 -> ~p6)
```

12.3.16 s4_s5_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_s5_n(i)), ", ", modaldepth(s4_s5_n(i)), ", ", vars(s4_s5_n(i)));
  1:  13, 3, [p1, p6]
  2:  85, 8, [p1, p2, p3, p4, p5, p6, p7, p12]
  3:  157, 14, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p18]
  4:  229, 20, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p24]
  5:  301, 26, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25,
p30]
  6:  373, 32, [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25,
p26, p27, p28, p29, p30, p31, p36]
> s4_s5_n(1);
  box dia (box p6 v false) v box (dia p1 -> ~p6)
> s4_s5_n(2);
  box dia (box p12 v (dia (p1 & ~p2 v ~p1 & p2) v box (dia (p2 & ~p3 v ~p2 & p3) v box (dia (p3 & ~p4 v ~p3 & p4) v box (dia (p4
& ~p5 v ~p4 & p5) v box (dia (p5 & ~p6 v ~p5 & p6) v box (dia (p6 & ~p7 v ~p6 & p7) v box false)))))) v box (dia p1 -> ~p12)
```

12.3.17 s4_t4p_p

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_t4p_p(i)), ", ", modaldepth(s4_t4p_p(i)), ", ", vars(s4_t4p_p(i)));
  1:  69, 5, [p0, p1, p3, p4]
  2:  127, 7, [p0, p1, p3, p4]
  3:  185, 9, [p0, p1, p3, p4]
  4:  243, 11, [p0, p1, p3, p4]
  5:  301, 13, [p0, p1, p3, p4]
  6:  359, 15, [p0, p1, p3, p4]
> s4_t4p_p(1);
  dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1) v (box (dia (p1 & dia ~p1) v p1) & (dia box p1 &
~p1) v dia (dia (dia box dia p0 & p0 & dia (~p0 v box p3)) v dia box (~p0 v box p3))) v dia p4
> s4_t4p_p(2);
  dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1)) v box (~box
p1 -> box ~box p1) v (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1)
v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (dia (dia box dia p0 & p0 & dia (~p0 v box p3)) v dia box (~p0 v box
p3))))) v dia p4
```

12.3.18 s4_t4p_n

```
> for i := 1 to 6 do
    print(i, ":  ", length(s4_t4p_n(i)), ", ", modaldepth(s4_t4p_n(i)), ", ", vars(s4_t4p_n(i)));
  1:  108, 7, [p0, p1, p3, p4]
  2:  224, 11, [p0, p1, p3, p4]
  3:  340, 15, [p0, p1, p3, p4]
  4:  456, 19, [p0, p1, p3, p4]
  5:  572, 23, [p0, p1, p3, p4]
  6:  688, 27, [p0, p1, p3, p4]
> s4_t4p_n(1);
  dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box (~box p1 -> box ~box p1) v (box (dia (p1 & dia ~p1) v p1) & (dia box p1 &
~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box
dia p0 & box (~p0 v box p3) v dia (dia box dia p0 & p0 & dia (~p0 v box p3)))))) v dia p4
> s4_t4p_n(2);
  dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box (dia ~(box ~p1 -> box box ~p1) v box dia box p0 v box
(~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1)) v box (~box p1 -> box ~box p1) v (box (dia (p1 & dia ~p1) v p1) & (dia box
p1 & ~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia
(box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box (dia (p1 &
dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box (dia (p1 & dia ~p1) v p1) & (dia box p1 & ~p1) v dia (box dia p0 & box (~p0 v box p3)
v dia (dia box dia p0 & p0 & dia (~p0 v box p3)))))))))) v dia p4
```

13 Appendix II: LWB programs that generate the formulas

13.1 Preliminary remarks

This appendix contains procedures that generate the formulas. The procedures are implemented in the LWB programming language (cp. LWB home page, item `programming language`).

In order to avoid inconsistencies between the formulas in the sections 6, 7, 8 and the programs in this appendix, we implemented a Perl program that automatically converts the LaTeX definitions into LWB programs. We then compared the results of these generated LWB programs with the results of the LWB programs in this appendix.

A typical session where such a procedure is used looks as follows:

```
 LWB  -  The Logics Workbench  1.0
 type 'help;' for help

> read("k_lin_p.lwb");
k  user
k> provable(k_lin_p(3));
true
k> quit
```

13.2 `fml_lib.lwb`

```
# fml_lib.lwb

####################################################################################################

load(k);

####################################################################################################

D     :== box p0 -> dia p0;
D2    :== dia true;
B     :== p0 -> box dia p0;
T     :== box p0 -> p0;
A4    :== box p0 -> box box p0;
A5    :== ~ box p0 -> box ~ box p0;
H     :== box(p0 v p1) & box(box p0 v p1) & box(p0 v box p1) -> box p0 v box p1;
L     :== box(p0 & box p0 -> p1) v box(p1 & box p1 -> p0);
Lplus :== box(box p0 -> p1) v box(box p1 -> p0);
Grz   :== box(box(p0 -> box p0) -> p0) -> p0;
Grz1  :== box(box(p0 -> box p0) -> p0) -> box p0;
Dum   :== box(box(p0 -> box p0) -> p0) -> (dia box p0 -> p0);
Dum1  :== box(box(p0 -> box p0) -> p0) -> (dia box p0 -> box p0);
Dum4  :== box(box(p0 -> box p0) -> p0) -> (dia box p0 -> p0 v box p0);

####################################################################################################

# Puts  n  box  in front of the formula  a .
# If  n  is 0 then  a  is returned.

proc : mbox(n, a)

local i;

begin
  for i := 1 to n do a := box a;
  return a;
end;

####################################################################################################

# Puts  n  dia  in front of the formula  a .
# If  n  is 0 then  a  is returned.

proc : mdia(n, a)

local i;

begin
  for i := 1 to n do a := dia a;
  return a;
end;
```

```
#####################################################################################################

proc : p(i)

begin
  return symbol("p", i);
end;

#####################################################################################################

# Converts the list  l  into a conjunction.

proc : list2conj(l)

local c, i, x;

begin
  if nops(l) = 0 then return true;
  c := l[1]; for i := 2 to nops(l) do c := c & l[i];
  return c;
end;

#####################################################################################################

# Converts the list  l  into a disjunction.

proc : list2disj(l)

local d, i, x;

begin
  if nops(l) = 0 then return false;
  d := l[1]; for i := 2 to nops(l) do d := d v l[i];
  return d;
end;

#####################################################################################################
#####################################################################################################
```

13.3 K

13.3.1 k_branch_p.lwb

```
# k_branch_p.lwb

#####################################################################################################

read("fml_lib.lwb");

#####################################################################################################

proc : k_branch_p(n)

local b, i, l;

begin
  b := bdepth(n) & det(n) & branching(n);
  l := []; for i := 0 to n do append(l, mbox(i, b));
  return ~(p(100) & ~p(101) & list2conj(l)) v ~mbox(n, p(n div 3 + 1));
end; # k_branch_p

#####################################################################################################

proc : bdepth(n)

local c, i;

begin
  c := []; for i := 1 to n + 1 do append(c, p(100 + i) -> p(99 + i));
  return list2conj(c);
end; # bdepth

#####################################################################################################

proc : det(n)

local c, i;

begin
  c := [];
  for i := 0 to n do
    append(c, p(100 + i)
              -> ((p(i) -> box(p(100 + i) -> p(i))) & (~p(i) -> box(p(100 + i) -> ~p(i)))));
  return list2conj(c);
end; # det
```

```
####################################################################################################

proc : branching(n)

local c, i;

begin
  c := [];
  for i := 0 to n - 1 do
    append(c, p(100 + i) & ~p(101 + i)
              -> (dia(p(101 + i) & ~p(102 + i) & p(i + 1)) & dia(p(101 + i) & ~p(102 + i) & ~p(i + 1))));
  return list2conj(c);
end; # branching

####################################################################################################
####################################################################################################
```

13.3.2 k_branch_n

```
# k_branch_n.lwb

####################################################################################################

read("k_branch_p.lwb");

####################################################################################################

proc : k_branch_n(n)

local b, i, l;

begin
  b := bdepth(n) & det(n) & branching(n);
  l := []; for i := 0 to n do append(l, mbox(i, b));
  return ~(p(100) & ~p(101) & list2conj(l));
end; # k_branch_n

####################################################################################################
####################################################################################################
```

13.3.3 k_d4_p

```
# k_d4_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_d4_p(n)

local i, d;

begin
  d := [];
  for i := 1 to n do
    append(d,
           mbox(n, T) v ~mbox(i, D2) v ~mbox(i, A4)
           v ~mbox(i, A4{dia p0/p0}) v ~mbox(i, B) v ~mbox(i, B{~p0/p0}));
  return k::nnf(list2disj(d));
end; # k_d4_p

####################################################################################################
####################################################################################################
```

13.3.4 k_d4_n

```
# k_d4_n.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_d4_n(n)

local i, d;

begin
  d := [];
```

```
  for i := 1 to n do
    d := concat(d,
                [mbox(n, box p0 v box dia ~p0)
                 v ~mbox(i, D2)
                 v ~mbox(i, A4)
                 v ~mbox(i, A4{dia p0/p0}) v ~mbox(i, D)
                 v ~mbox(i, A4{dia p0 -> p0/p0})
                 v ~mbox(i, A4{box p0 -> p0/p0})]);
  return k::nnf(list2disj(d));
end; # k_d4_n

####################################################################################################
####################################################################################################
```

13.3.5 k_dum_p

```
# k_dum_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_dum_p(n)

local c, d1, d2;

begin
  c := A4{box(p0 -> box p0) -> p0 / p0} & box A4 & Dum & Dum{p0 -> box p0 / p0};
  d1 := []; for i := 1 to n div 2 do append(d1, mbox(i, box A4 & Dum));
  d2 := []; for i := n div 2 + 2 to n - 1 do append(d2, mdia(i, ~(box A4 & Dum)));
  return list2conj(d1) & ~ mbox(n div 2 + 1, Dum1) -> mdia(n div 2 + 1, ~c) v list2disj(d2);
end; # k_dum_p

####################################################################################################
####################################################################################################
```

13.3.6 k_dum_n

```
# k_dum_n.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_dum_n(n)

local c, d1, d2, i;

begin
  c := A4{box(p0 -> box p0) -> p0 / p0} & box A4 & Dum4 & Dum4{p0 -> box p0 / p0};
  d1 := []; for i := 1 to n div 2 do append(d1, mbox(i, box A4 & Dum4));
  d2 := []; for i := n div 2 + 2 to n - 1 do append(d2, mdia(i, ~(box A4 & Dum4)));
  return list2conj(d1) & ~ mbox(n + 1, Dum) -> mdia(n + 1, ~c) v list2disj(d2);
end; # k_dum_n

####################################################################################################
####################################################################################################
```

13.3.7 k_grz_p

```
# k_grz_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_grz_p(n)

local c, i, d;

begin
  C := box(p2 -> box p2) -> p2;
  d := []; for i := 1 to n - 1 do append(d, l(i));
  return box Grz{p2/p0} & list2conj(d) & Grz{C & A4{C/p0}/p0} -> Grz1{p1/p0} v Grz1{p2/p0} v Grz1{p3/p0};
end; # k_grz_p
```

```
##################################################################################################

proc : l(i)

begin
  if i mod 4 = 0 then return Grz{l2(i div 4) / p0};
  if i mod 4 = 1 then return Grz{box l2(i div 4) v p1 / p0};
  if i mod 4 = 2 then return Grz{box l2(i div 4) v p1 v p2 / p0};
  return Grz{box l2(i div 4) v p1 v p2 v p3 / p0};
end; # l

##################################################################################################

proc : l2(i)

begin
  if i = 0 then return false; else return box l2(i - 1) v p1 v p2 v p3 v p4;
end; # l2

##################################################################################################
##################################################################################################
```

13.3.8 k_grz_n

```
# k_grz_n.lwb

##################################################################################################

read("k_grz_p.lwb");

##################################################################################################

proc : k_grz_n(n)

local c, i, d;

begin
  C := box(p2 -> box p2) -> p2;
  d := []; for i := 1 to n - 1 do append(d, l(i));
  return box Grz1{p2/p0} & list2conj(d) & Grz1{C & A4{C/p0}/p0} -> Grz{p1/p0} v Grz{p2/p0} v Grz{p3/p0};
end; # k_grz_n

##################################################################################################

proc : l(i)

begin
  if i mod 4 = 0 then return Grz1{l2(i div 4) / p0};
  if i mod 4 = 1 then return Grz1{box l2(i div 4) v p1 / p0};
  if i mod 4 = 2 then return Grz1{box l2(i div 4) v p1 v p2 / p0};
  return Grz1{box l2(i div 4) v p1 v p2 v p3 / p0};
end; # l

##################################################################################################
##################################################################################################
```

13.3.9 k_lin_p

```
# k_lin_p.lwb

##################################################################################################

read("fml_lib.lwb");

##################################################################################################

proc : k_lin_p(n)

local d1, d2, i, H2;

begin
  H2 := H{p0 & box p0 & p0 -> p1/p0, ~p0 -> ~(box p1 & p1)/p1};
  d1 := []; for i := 1 to n div 3 do append(d1, ~ H2{p(i)/p0, p(i + 1)/p1});
  d2 := []; for i := ((n div 3) + 1) to n do append(d2, ~ H2{p(i)/p0, p(i + 1)/p1});
  return list2disj(d1) v L{p(n)/p0, p(n)/p1} v list2disj(d2);
end; # k_lin_p

##################################################################################################
##################################################################################################
```

13.3.10 k_lin_n

```
# k_lin_n.lwb

#########################################################################################################

read("fml_lib.lwb");

#########################################################################################################

proc : k_lin_n(n)

local d1, d2, i;

begin
  d1 := [];
  for i := 1 to 2 mult n - 2 do
    append(d1, ~ L{dia p(i)/p0, p(i + 1)/p1} v ~ L{p(i) -> box p(i + 1)/p0, p(i + 1)/p1});
  d2 := [];
  for i := 2 mult n to 4 mult n - 4 do
    append(d2, ~ L{dia p(i)/p0, p(i + 1)/p1} v ~ L{p(i) -> box p(i + 1)/p0, p(i + 1)/p1});
  return list2disj(d1) v Lplus{p(n)/p0, p(n)/p1} v list2disj(d2);
end; # k_lin_n

#########################################################################################################
#########################################################################################################
```

13.3.11 k_path_p

```
# k_path_p.lwb

#########################################################################################################

read("fml_lib.lwb");

#########################################################################################################

proc : k_path_p(n)

local d1, d2, d3, i, j;

begin
  d1 := []; foreach i in [1,path_el(1,n),3,5] do append(d1, box p(i));
  d2 := [];
  for i := 1 to n do
    for j := 1 to 6 do
      append(d2, (left_to_right(i, j, n) v right_to_left(i, j, n)));
  d3 := []; foreach i in [2,4,path_el(n,n),6] do append(d3, mdia(n, ~p(i)));
  return list2disj(d1) v list2disj(d2) v list2disj(d3);
end; # k_path_p

#########################################################################################################

proc : path_el(level, n)

local x;

begin
  if (level = 1) then return 2;
  if (level > n div 2) then x := path_el(level - 1, n) + 3; else x := path_el(level - 1, n) + 5;
  if (x > 6) then x := x - 6;
  return x;
end; # path_el

#########################################################################################################

proc : left_to_right(level, k, n)

local x;

begin
  if level = n then return false;
  if (k mod 2 = 0) and (k <> path_el(level,n)) then return false;
  x := path_el(level + 1, n);
  if (k mod 2 = 0) then return lists2fml(level, k, [x]);
  if (k = path_el(level,n)) then return lists2fml(level, k, [1,3,x,5]);
  return lists2fml(level, k, delete(x, 1, 3, 5));
end; # left_to_right

#########################################################################################################

proc : right_to_left(level, k, n)

local x;

begin
  if level = 1 then return false;
```

```
  if (k mod 2 = 1) then return false;
  x := path_el(level - 1, n);
  return lists2fml_back(level - 1, k, delete(x, 2, 4, 6));
end; # right_to_left

############################################################################################

proc : delete(x, y1, y2, y3)

begin
  if x = y1 then return [y2,y3];
  if x = y2 then return [y1,y3];
  if x = y3 then return [y1,y2];
  return [y1,y2,y3];
end; # delete

############################################################################################

proc : lists2fml(level, k, s)

local d, x;

begin
  d := []; foreach x in s do append(d, mdia(level, ~p(k) & box p(x)));
  return list2disj(d);
end; # lists2fml

############################################################################################

proc : lists2fml_back(level, k, s)

local d, x;

begin
  d := []; foreach x in s do append(d, mdia(level, ~p(x) & box p(k)));
  return list2disj(d);
end; # lists2fml_back

############################################################################################
############################################################################################
```

13.3.12 k_path_n

```
# k_path_p.lwb

############################################################################################

read("fml_lib.lwb");

############################################################################################

proc : k_path_p(n)

local d1, d2, d3, i, j;

begin
  d1 := []; foreach i in [1,path_el(1,n),3,5] do append(d1, box p(i));
  d2 := [];
  for i := 1 to n do
    for j := 1 to 6 do
      append(d2, (left_to_right(i, j, n) v right_to_left(i, j, n)));
  d3 := []; foreach i in [2,4,path_el(n,n),6] do append(d3, mdia(n, ~p(i)));
  return list2disj(d1) v list2disj(d2) v list2disj(d3);
end; # k_path_p

############################################################################################

proc : path_el(level, n)

local x;

begin
  if (level = 1) then return 2;
  if (level > n div 2) then x := path_el(level - 1, n) + 3; else x := path_el(level - 1, n) + 5;
  if (x > 6) then x := x - 6;
  return x;
end; # path_el

############################################################################################

proc : left_to_right(level, k, n)

local x;

begin
  if level = n then return false;
  if (k mod 2 = 0) and (k <> path_el(level,n)) then return false;
```

```
  x := path_el(level + 1, n);
  if (k mod 2 = 0) then return lists2fml(level, k, [x]);
  if (k = path_el(level,n)) then return lists2fml(level, k, [1,3,x,5]);
  return lists2fml(level, k, delete(x, 1, 3, 5));
end; # left_to_right

####################################################################################################

proc : right_to_left(level, k, n)

local x;

begin
  if level = 1 then return false;
  if (k mod 2 = 1) then return false;
  x := path_el(level - 1, n);
  return lists2fml_back(level - 1, k, delete(x, 2, 4, 6));
end; # right_to_left

####################################################################################################

proc : delete(x, y1, y2, y3)

begin
  if x = y1 then return [y2,y3];
  if x = y2 then return [y1,y3];
  if x = y3 then return [y1,y2];
  return [y1,y2,y3];
end; # delete

####################################################################################################

proc : lists2fml(level, k, s)

local d, x;

begin
  d := []; foreach x in s do append(d, mdia(level, ~p(k) & box p(x)));
  return list2disj(d);
end; # lists2fml

####################################################################################################

proc : lists2fml_back(level, k, s)

local d, x;

begin
  d := []; foreach x in s do append(d, mdia(level, ~p(x) & box p(k)));
  return list2disj(d);
end; # lists2fml_back

####################################################################################################
####################################################################################################
```

13.3.13 k_ph_p

```
# k_ph_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_ph_p(n)

begin
  return dia left(n) -> dia right(n);
end; # k_ph_p

####################################################################################################

proc : left(n)

local c, d, j, i;

begin
  c := [];
  for i := 1 to n + 1 do
  begin
    d := []; for j := 1 to n do append(d, l(i, j));
    append(c, list2disj(d));
  end;
  return list2conj(c);
end; # left
```

```
####################################################################################################

proc : right(n)

local d, j, i1, i2;

begin
  d := [];
  for j := 1 to n do
    for i1 := 1 to n + 1 do
      for i2 := i1 + 1 to n + 1 do
        append(d, l(i1, j) & l(i2, j));
  return list2disj(d);
end; # right

####################################################################################################

proc : l(i, j)

begin
  if i < j then return box p(i mult 100 + j); else return p(i mult 100 + j);
end; # l

####################################################################################################
####################################################################################################
```

13.3.14 k_ph_n

```
# k_ph_n.lwb

####################################################################################################

read("k_ph_p.lwb");

####################################################################################################

proc : k_ph_n(n)

begin
  return dia left(n) -> dia right(n);
end; # k_ph_n

####################################################################################################

proc : right(n)

local d, j, i1, i2;

begin
  d := [];
  for j := 1 to n do
    for i1 := 1 to n + 1 do
      for i2 := i1 + 1 to n + 1 do
        append(d, l2(n, i1, j) & l2(n, i2, j));
  return list2disj(d);
end; # right

####################################################################################################

proc : l2(n, i, j)

begin
  if ( i = j ) and ( i = (2 mult n) div 3 + 1) then return ~l(i, j); else return l(i, j);
end; # l2

####################################################################################################
####################################################################################################
```

13.3.15 k_poly_p

```
# k_poly_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_poly_p(n)

begin
  if (n mod 2 = 1) then return poly(3 mult n); else return poly(3 mult n + 1);
end; # k_poly_p

####################################################################################################
```

```
proc : poly(n)

local c1, c2, i;

begin
  c1 := []; for i := 1 to n + 1 do append(c1, p(i));
  c2 := []; for i := 1 to n + 1 do append(c2, ~p(2 mult i));
  return mbox(n + 1, list2conj(c1)) v f(n,n) v mbox(n + 1, list2conj(c2));
end; # poly

####################################################################################################

proc : f(i, n)

local a, j;

begin
  if (i = 0) then return false;
  if (i = n) then return dia(f(n - 1, n) v mdia(n, p(n) <-> p(1))) v box p(n + 2);
  return dia(f(i - 1,n) v mdia(i, p(i) <-> p(i + 1))) v box p(i + 2);
end; # f

####################################################################################################
####################################################################################################
```

13.3.16 k_poly_n

```
# k_poly_n.lwb

####################################################################################################

read("k_poly_p.lwb");

####################################################################################################

proc : k_poly_n(n)

begin
  if (n mod 2 = 0) then return poly(3 mult n); else return poly(3 mult n + 1);
end; # k_poly_n

####################################################################################################
####################################################################################################
```

13.3.17 k_t4p_p

```
# k_t4p_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : k_t4p_p(n)

begin
  return E(n) v nnf(~C(n)) v dia p4;
end; # k_t4p_p

####################################################################################################

proc : C(i)

begin
  if i = 0
  then return ((box dia p0 -> dia p1)
              & box(box ~ box dia p1 -> ~ box dia p0)
              & (box dia p0 -> box box dia p1)
              & box(dia box dia p0 & p0 -> box p1)
              & box dia p1){p0 & dia ~p3/p1};

  return box A4{p1/p0} & box C(i - 1) & box A4{dia p1/p0};
end; # C

####################################################################################################

proc : E(i)

begin
  if i = 0 then return dia box p0;
  return dia ~ A4{~p1/p0} v box E(i - 1) v box A5{p1/p0};
end; # E
```

```
####################################################################################################
####################################################################################################
```

13.3.18 k_t4p_n

```
# k_t4p_n.lwb

####################################################################################################

read("k_t4p_p.lwb");

####################################################################################################

proc : k_t4p_n(n)

begin
  return E(2 mult n - 1) v nnf(~C(2 mult n - 1)) v dia p4;
end; # k_t4p_n

####################################################################################################

proc : C(i)

begin
  if i = 0
  then return ((box dia p0 -> dia p1)
               & box(box ~ box dia p1 -> ~ box dia p0)
               & (box dia p0 -> box box dia p1)
               & box(dia box dia p0 & p0 -> box p1)){p0 & dia ~p3/p1};
  return box A4{p1/p0} & box C(i - 1) & box A4{dia p1/p0};
end; # C

####################################################################################################
####################################################################################################
```

13.4 KT

13.4.1 kt_45_p

```
# kt_45_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : kt_45_p(n)

local i, d;

begin
  d := [];
  for i := 1 to n do
    append(d, mbox(n, A4) v ~mbox(i, D2)
              v ~mbox(i, A5{dia ~p0/p0}) v ~mbox(i, box A5) v ~mbox(i, B));
  return k::nnf(list2disj(d));
end; # kt_45_p

####################################################################################################
####################################################################################################
```

13.4.2 kt_45_n

```
# kt_45_n.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : kt_45_n(n)

local i, d;

begin
  d := [];
  for i := 1 to n do
```

```
    append(d, mbox(n, box p0 v box dia ~p0) v ~mbox(i, A4)
              v ~mbox(i, A4{dia p0/p0}) v ~mbox(i, T)
              v ~mbox(i, A4{dia p0 -> p0/p0})
              v ~mbox(i, A4{box p0 -> p0/p0}));
  return k::nnf(list2disj(d));
end; # kt_45_n

####################################################################################################
####################################################################################################
```

13.4.3 kt_branch_p

```
# kt_branch_p.lwb

####################################################################################################

read("k_branch_p.lwb");

####################################################################################################

proc : kt_branch_p(n)

begin
  return ~(p(100) & ~p(101) & mbox(n, bdepth(n) & det(n) & branching(n))) v ~mbox(n, p(n div 3 + 1));
end; # kt_branch_p

####################################################################################################
####################################################################################################
```

13.4.4 kt_branch_n

```
# kt_branch_n.lwb

####################################################################################################

read("k_branch_p.lwb");

####################################################################################################

proc : kt_branch_n(n)

begin
  return ~(p(100) & ~p(101) & mbox(n, bdepth(n) & det(n) & branching(n)));
end; # kt_branch_n

####################################################################################################
####################################################################################################
```

13.4.5 kt_dum_p

```
# kt_dum_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : kt_dum_p(n)

local c, d1, d2;

begin
  c := A4{box(p0 -> box p0) -> p0 / p0} & box A4 & Dum & Dum{p0 -> box p0 / p0};
  d1 := []; for i := 1 to n div 2 do append(d1, mbox(i, A4));
  d2 := []; for i := n div 2 + 2 to n - 1 do append(d2, mdia(i, ~A4));
  return list2conj(d1) & ~ mbox(n div 2 + 1, Dum1) -> mdia(n div 2 + 1, ~c) v list2disj(d2);
end; # kt_dum_p

####################################################################################################
####################################################################################################
```

13.4.6 kt_dum_n

```
# kt_dum_n.lwb

####################################################################################################

read("fml_lib.lwb");
```

```
####################################################################################################

proc : kt_dum_n(n)

local c, d1, d2, i;

begin
  c := A4{box(p0 -> box p0) -> p0 / p0} & box A4 & Dum & Dum{p0 -> box p0 / p0};
  d1 := []; for i := 1 to n div 2 do append(d1, mbox(i, box A4 & Dum));
  d2 := []; for i := n div 2 + 2 to n - 1 do append(d2, mdia(i, ~(box A4 & Dum)));
  return list2conj(d1) & ~ mbox(n div 2 + 1, Grz) -> mdia(n div 2 + 1, ~c) v list2disj(d2);
end; # kt_dum_n

####################################################################################################
####################################################################################################
```

13.4.7 kt_grz_p

```
# kt_grz_p.lwb

####################################################################################################

read("k_grz_p.lwb");

####################################################################################################

proc : kt_grz_p(n)

local a, i, d;

begin
  C := box(p2 -> box p2) -> p2;
  d := []; for i := 1 to n - 1 do append(d, l(i));
  return box Grz{p2/p0} & list2conj(d) & Grz{C & A4{C/p0}/p0}
         -> Grz1{p1/p0} v Grz1{p2/p0} & dia box ~p0 v Grz1{p3/p0} v Grz1{p2/p0} & dia p0;
end; # kt_grz_p

####################################################################################################
####################################################################################################
```

13.4.8 kt_grz_n

```
# kt_grz_n.lwb

####################################################################################################

read("kt_grz_p.lwb");

####################################################################################################

proc : kt_grz_n(n)

local c, i, d;

begin
  C := box(p2 -> box p2) -> p2;
  d := []; for i := 1 to n - 1 do append(d, l(i));
  return box Grz1{p2/p0} & list2conj(d) & Grz1{C & A4{C/p0}/p0} -> A5{p1/p0} v A5{p2/p0} v A5{p3/p0};
end; # kt_grz_n

####################################################################################################

proc : l(i)

begin
  if i mod 4 = 0 then return Grz1{l2(i div 4) / p0};
  if i mod 4 = 1 then return Grz1{box l2(i div 4) v p1 / p0};
  if i mod 4 = 2 then return Grz1{box l2(i div 4) v p1 v p2 / p0};
  return Grz1{box l2(i div 4) v p1 v p2 v p3 / p0};
end; # l

####################################################################################################
####################################################################################################
```

13.4.9 kt_md_p

```
# kt_md_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################
```

```
proc : kt_md_p(n)

local d, i;

begin
  d := [];
  for i := 1 to n - 1 do
    append(d, g(i, n, ~p1 & dia f(1, n, p2))
              v g(i, n, ~p1 & dia f(1, n, p1))
              v g(i, n, ~p2 & dia f(1, n, p1)));
  return p1 v list2disj(d) v g(n, n, ~p1);
end; # kt_md_p

####################################################################################################

proc : g(i, n, a)

local j;

begin
  if i = 1 then return a;
  return f(i, n, g(i - 1, n, a));
end; # g

####################################################################################################

proc : f(i, n, a)

local j;

begin
  return mdia(i - 1, box mdia(n - i, a));
end; # f

####################################################################################################
####################################################################################################
```

13.4.10 kt_md_n

```
# kt_md_n.lwb

####################################################################################################

read("kt_md_p.lwb");

####################################################################################################

proc : kt_md_n(n)

local d, i;

begin
  d := []; for i := 1 to n - 1 do append(d, g(i, n, ~p1 & dia f(1, n, p1)));
return p1 v list2disj(d);

end; # kt_md_n

####################################################################################################
####################################################################################################
```

13.4.11 kt_path_p

```
# kt_path_p.lwb

####################################################################################################

read("k_path_p.lwb");

####################################################################################################

proc : kt_path_p(n)

local d1, d2, d3, i, j;

begin
  d1 := []; foreach i in [1,path_el(1,n),3,5] do append(d1, box p(10 + i));
  d2 := [];
  for i := 1 to n do
    for j := 1 to 6 do
      append(d2, (left_to_right(i, j, n) v right_to_left(i, j, n)));
  d3 := []; foreach i in [2,4,path_el(n,n),6] do append(d3, mdia(n, ~p(10 mult n + i)));
  return list2disj(d1) v list2disj(d2) v list2disj(d3);
end; # kt_path_p
```

```
####################################################################################################

proc : lists2fml(level, k, s)

local d, x;

begin
  d := [];
  foreach x in s do append(d, mdia(level, ~p(10 mult level + k) & box p(10 mult (level + 1) + x)));
  return list2disj(d);
end; # lists2fml

####################################################################################################

proc : lists2fml_back(level, k, s)

local d, x;

begin
  d := [];
  foreach x in s do append(d, mdia(level, ~p(10 mult level + x) & box p(10 mult (level + 1) + k)));
  return list2disj(d);
end; # lists2fml_back

####################################################################################################
####################################################################################################
```

13.4.12 kt_path_n

```
# kt_path_n.lwb

####################################################################################################

read("k_path_n.lwb");

####################################################################################################

proc : kt_path_n(n)

local d1, d2, d3, i, j;

begin
  d1 := []; foreach i in [1,path_el(1,n+1),3,5] do append(d1, box p(10 + i));
  d2 := [];
  for i := 1 to n + 1 do
    for j := 1 to 6 do
      append(d2, (left_to_right(i, j, n + 1) v right_to_left(i, j, n + 1)));
  d3 := [];
  foreach i in [2,4,path_el(n+1,n+1),6] do append(d3, mdia(n + 1, ~p(10 mult (n + 1) + i)));
  return list2disj(d1) v list2disj(d2) v list2disj(d3);
end; # kt_path_n

####################################################################################################

proc : lists2fml(level, k, s)

local d, x;

begin
  d := [];
  foreach x in s do append(d, mdia(level, ~p(10 mult level + k) & box p(10 mult (level + 1) + x)));
  return list2disj(d);
end; # lists2fml

####################################################################################################

proc : lists2fml_back(level, k, s)

local d, x;

begin
  d := [];
  foreach x in s do append(d, mdia(level, ~p(10 mult level + x) & box p(10 mult (level + 1) + k)));
  return list2disj(d);
end; # lists2fml_back

####################################################################################################
####################################################################################################
```

13.4.13 kt_ph_p

```
# kt_ph_p.lwb
```

```
####################################################################################################

read("k_ph_p.lwb");

####################################################################################################

proc : kt_ph_p(n)

begin
  return left(n) -> dia right(n);
end; # kt_ph_p

####################################################################################################
####################################################################################################
```

13.4.14 kt_ph_n

```
# kt_ph_n.lwb

####################################################################################################

read("k_ph_n.lwb");

####################################################################################################

proc : kt_ph_n(n)

begin
  return left(n) -> dia right(n);
end; # kt_ph_n

####################################################################################################
####################################################################################################
```

13.4.15 kt_poly_p

```
# kt_poly_p.lwb

####################################################################################################

read("k_poly_p.lwb");

####################################################################################################

proc : kt_poly_p(n)

begin
  if (n mod 2 = 1) then return poly(5 mult n); else return poly(5 mult n + 1);
end; # kt_poly_p

####################################################################################################

proc : f(i, n)

local a, j;

begin
  if (i = 0) then return false;
  if (i = n) then return dia(f(n - 1, n) v mdia(n + 2, p(n) <-> p(1))) v box p(n + 2);
  return dia(f(i - 1,n) v mdia(i + 2, p(i) <-> p(i + 1))) v box p(i + 2);
end; # f

####################################################################################################
####################################################################################################
```

13.4.16 kt_poly_n

```
# kt_poly_n.lwb

####################################################################################################

read("kt_poly_p.lwb");

####################################################################################################

proc : kt_poly_n(n)

begin
  if (n mod 2 = 0) then return poly(3 mult n); else return poly(3 mult n + 1);
end; # kt_poly_n

####################################################################################################
####################################################################################################
```

13.4.17 kt_t4p_p

```
# kt_t4p_p.lwb

############################################################################################################

read("k_t4p_p.lwb");

############################################################################################################

proc : kt_t4p_p(n)

begin
  return E(n) v nnf(~C(n)) v dia p4;
end; # kt_t4p_p

############################################################################################################

proc : C(i)

begin
  if i = 0
  then return ((box dia p0 -> box box dia p1)
             & box(dia box dia p0 & p0 -> box p1)
             & box dia p1){p0 & dia ~p3/p1};
  return box A4{p1/p0} & box C(i - 1);
end; # C

############################################################################################################
############################################################################################################
```

13.4.18 kt_t4p_n

```
# kt_t4p_n.lwb

############################################################################################################

read("kt_t4p_p.lwb");

############################################################################################################

proc : kt_t4p_n(n)

begin
  return E(2 mult n - 1) v nnf(~C(2 mult n - 1)) v dia p4;
end; # kt_t4p_n

############################################################################################################

proc : C(i)

begin
  if i = 0
  then return ((box dia p0 -> dia p1)
             & (box dia p0 -> box box dia p1)
             & box(dia box dia p0 & p0 -> box p1)){p0 & dia ~p3/p1};
  return box A4{p1/p0} & box C(i - 1);
end; # C

############################################################################################################
############################################################################################################
```

13.5 S4

13.5.1 s4_45_p

```
# s4_45_p.lwb

############################################################################################################

read("fml_lib.lwb");

############################################################################################################

proc : s4_45_p(n)

local i, d;

begin
  d := [];
  for i := 1 to n do
    append(d, h(n, A5) v ~h(i, D2) v ~h(i, A5{dia ~p0/p0}) v ~h(i, box A5) v ~h(i, B));
  return k::nnf(list2disj(d));
end; # s4_45_p
```

```
####################################################################################################

proc : h(i, a)

begin
  if i = 0 then return a; else return box p0 v box h(i - 1, a) v box p1;
end;

####################################################################################################
####################################################################################################
```

13.5.2 s4_45_n

```
# s4_45_n.lwb

####################################################################################################

read("s4_45_p.lwb");

####################################################################################################

proc : s4_45_n(n)

local i, d;

begin
  d := [];
  for i := 1 to n do
    append(d, h(n, box p0 v box dia ~p0) v ~h(i, A4)
              v ~h(i, A4{dia p0/p0}) v ~h(i, T)
              v ~h(i, A4{box p0 -> p0/p0})
              v ~h(i, T{box p0 -> p0/p0}));
  return k::nnf(list2disj(d));
end; # s4_45_n

####################################################################################################
####################################################################################################
```

13.5.3 s4_branch_p

```
# s4_branch_p.lwb

####################################################################################################

read("k_branch_p.lwb");

####################################################################################################

proc : s4_branch_p(n)

begin
  return ~(p(100) & ~p(101) & box(bdepth(n) & det(n) & branching(n))) v ~box p(n div 3 + 1);
end; # s4_branch_p

####################################################################################################
####################################################################################################
```

13.5.4 s4_branch_n

```
# s4_branch_n.lwb

####################################################################################################

read("k_branch_p.lwb");

####################################################################################################

proc : s4_branch_n(n)

begin
  return ~(p(100) & ~p(101) & box(bdepth(n) & det(n) & branching(n)));
end; # s4_branch_n

####################################################################################################
####################################################################################################
```

13.5.5 s4_grz_p

```
# s4_grz_p.lwb
```

```
####################################################################################################

read("k_grz_p.lwb");

####################################################################################################

proc : s4_grz_p(n)

local a, C, i, d1;

begin
  C := box(p2 -> box p2) -> p2;
  d1 := []; for i := 1 to n - 1 do append(d1, l(i));
  return box Grz{p2/p0} & list2conj(d1) & Grz{C & A4{C/p0}/p0}
         -> Grz1{p1/p0} v Grz1{p2/p0} & ~ box dia p0 v Grz1{p3/p0} v Grz1{p2/p0} & ~ dia ~(box dia p0 v p1);
end; # s4_grz_p

####################################################################################################
####################################################################################################
```

13.5.6 s4_grz_n

```
# s4_grz_n.lwb

####################################################################################################

read("s4_grz_p.lwb");

####################################################################################################

proc : s4_grz_n(n)

local C, i, d;

begin
  C := box(p2 -> box p2) -> p2;
  d := []; for i := 1 to n - 1 do append(d, l(i));
  return box Grz1{p2/p0} & list2conj(d) & Grz1{C & A4{C/p0}/p0} -> A5{p1/p0} v A5{p2/p0} v A5{p3/p0};
end; # s4_grz_n

####################################################################################################

proc : l(i)

begin
  if i mod 4 = 0 then return Grz1{l2(i div 4) / p0};
  if i mod 4 = 1 then return Grz1{box l2(i div 4) v p1 / p0};
  if i mod 4 = 2 then return Grz1{box l2(i div 4) v p1 v p2 / p0};
  return Grz1{box l2(i div 4) v p1 v p2 v p3 / p0};
end; # l

####################################################################################################
####################################################################################################
```

13.5.7 s4_ipc_p

```
# s4_ipc_p.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : s4_ipc_p(n)

local c, i;

begin
  c := []; for i := 1 to n do append(c, f(i, n));
  return list2conj(c) -> false;
end; # s4_ipc_p

####################################################################################################

proc : f(i, n)

local c, j;

begin
  c := []; for j := 1 to n do append(c, box p(j));
  return box(box(box p(i) -> list2conj(c)) -> false);
end; # f
```

```
####################################################################################################
####################################################################################################
```

13.5.8 s4_ipc_n

```
# s4_ipc_n.lwb

####################################################################################################

read("fml_lib.lwb");

####################################################################################################

proc : s4_ipc_n(n)

local c, i;

begin
  c := []; for i := 1 to n do append(c, f2(i, n));
  return list2conj(c) -> false;
end; # s4_ipc_n

####################################################################################################

proc : f2(i, n)

begin
  if i = (n + 1) div 2 then return true; else return f(i, n);
end; # f2

####################################################################################################
proc : f(i, n)

local c, j;

begin
  c := []; for j := 1 to n do append(c, box p(j));
  return box(box(box p(i) -> list2conj(c)) -> false);
end; # f

####################################################################################################
####################################################################################################
```

13.5.9 s4_md_p

```
# s4_md_p.lwb

####################################################################################################

read("kt_md_p.lwb");

####################################################################################################

proc : s4_md_p(n)

local d, i;

begin
  d := [];
  for i := 1 to n - 1 do
    append(d, g(i, n, ~p1 & dia f(1, n, p2))
              v g(i, n, ~p1 & dia f(1, n, p1))
              v g(i, n, ~p2 & dia f(1, n, p1)));
  return p1 v list2disj(d) v g(n, n, ~p1);
end; # s4_md_p

####################################################################################################

proc : f(i, n, a)

local j;

begin
  return h(i - 1, box h(n - i, a));
end; # f

####################################################################################################

proc : h(i, a)

local j;

begin
  if i = 0 then return a; else return dia h(i - 1, a) v p2;
end; # h
```

```
####################################################################################################
####################################################################################################
```

13.5.10 s4_md_n

```
# s4_md_n.lwb

####################################################################################################

read("s4_md_p.lwb");

####################################################################################################

proc : s4_md_n(n)

local d, i;

begin
  d := []; for i := 1 to n - 1 do append(d, g(i, n, ~p1 & dia f(1, n, p1)));
  return p1 v list2disj(d);
end; # s4_md_n

####################################################################################################
####################################################################################################
```

13.5.11 s4_path_p

```
# s4_path_p.lwb

####################################################################################################

read("kt_path_p.lwb");

####################################################################################################

proc : s4_path_p(n)

local d1, d2, d3, i, j;

begin
  d1 := []; foreach i in [1,path_el(1,n),3,5] do append(d1, box box p(10 + i));
  d2 := [];
  for i := 1 to n do
    for j := 1 to 6 do
      append(d2, (left_to_right(i, j, n) v right_to_left(i, j, n)));
  d3 := []; foreach i in [2,4,path_el(n,n),6] do append(d3, dia ~p(10 mult n + i));
  return list2disj(d1) v list2disj(d2) v dia list2disj(d3);
end; # s4_path_p

####################################################################################################
####################################################################################################
```

13.5.12 s4_path_n

```
# s4_path_n.lwb

####################################################################################################

read("kt_path_n.lwb");

####################################################################################################

proc : s4_path_n(n)

local d1, d2, d3, i, j;

begin
  d1 := []; foreach i in [1,path_el(1,n+1),3,5] do append(d1, box box p(10 + i));
  d2 := [];
  for i := 1 to n + 1 do
    for j := 1 to 6 do
      append(d2, (left_to_right(i, j, n + 1) v right_to_left(i, j, n + 1)));
  d3 := []; foreach i in [2,4,path_el(n+1,n+1),6] do append(d3, dia ~p(10 mult (n + 1) + i));
  return list2disj(d1) v list2disj(d2) v dia list2disj(d3);
end; # s4_path_n

####################################################################################################
####################################################################################################
```

13.5.13 s4_ph_p

```
# s4_ph_p.lwb

####################################################################################################

read("k_ph_p.lwb");

####################################################################################################

proc : s4_ph_p(n)

begin
  return left(n) -> dia right(n);
end; # s4_ph_p

####################################################################################################

proc : right(n)

local d, j, i1, i2;

begin
  d := [];
  for j := 1 to n do
    for i1 := 1 to n + 1 do
      for i2 := i1 + 1 to n + 1 do
        append(d, dia(l(i1, j) & l(i2, j)));
  return list2disj(d);
end; # right

####################################################################################################
####################################################################################################
```

13.5.14 s4_ph_n

```
# s4_ph_n.lwb

####################################################################################################

read("s4_ph_p.lwb");

####################################################################################################

proc : s4_ph_n(n)

begin

return left(n) -> dia right(n);

end; # s4_ph_n

####################################################################################################

proc : right(n)

local d, j, i1, i2;

begin

d := [];
for j := 1 to n do
  for i1 := 1 to n + 1 do
    for i2 := i1 + 1 to n + 1 do
      append(d, dia(l2(n, i1, j) & l2(n, i2, j)));

return list2disj(d);

end; # right

####################################################################################################

proc : l2(n, i, j)

begin

if ( i = j ) and ( i = (2 mult n) div 3 + 1)
then return ~l(i, j);
else return l(i, j);

end; # l2

####################################################################################################
####################################################################################################
```

13.5.15 s4_s5_p

```
# s4_s5_p.lwb

################################################################################################

read("fml_lib.lwb");

################################################################################################

proc : s4_s5_p(n)

local d, i;

begin
  d := []; for i := 1 to 3 mult n - 2 do append(d, p(i) <-> p(i + 1));
  return box dia(box list2disj(d) v box p(3 mult n) v f(1, 3 mult n - 1)) v box(dia p1 -> ~p(3 mult n));
end; # s4_s5_p

################################################################################################

proc : f(i, n)

begin
  if i = n then return false; else return dia(p(i) & ~p(i + 1) v ~p(i) & p(i + 1)) v box f(i + 1, n);
end;

################################################################################################
################################################################################################
```

13.5.16 s4_s5_n

```
# s4_s5_n.lwb

################################################################################################

read("s4_s5_p.lwb");

################################################################################################

proc : s4_s5_n(n)

begin

return box dia(box p(6 mult n) v f(1, 6 mult n - 5)) v box(dia p1 -> ~p(6 mult n));

end; # s4_s5_n

################################################################################################
################################################################################################
```

13.5.17 s4_t4p_p

```
# s4_t4p_p.lwb

################################################################################################

read("k_t4p_p.lwb");

################################################################################################

proc : s4_t4p_p(n)

begin
  return E(n) v nnf(~C(2 mult n - 1)) v dia p4;
end; # s4_t4p_p

################################################################################################

proc : C(i)

begin
  if i = 0 then return (box(dia box dia p0 & p0 -> box p1) & box dia p1){p0 & dia ~p3/p1};
  return Dum{p1/p0} & box C(i - 1);
end; # C

################################################################################################
################################################################################################
```

13.5.18 s4_t4p_n

```
# s4_t4p_n.lwb
```

```
####################################################################################################

read("s4_t4p_p.lwb");

####################################################################################################

proc : s4_t4p_n(n)

begin
  return E(2 mult n - 1) v nnf(~C(4 mult n - 1)) v dia p4;
end; # s4_t4p_n

####################################################################################################

proc : C(i)

begin
  if i = 0 then return ((box dia p0 -> dia p1) & box(dia box dia p0 & p0 -> box p1)){p0 & dia ~p3/p1};
  return Dum{p1/p0} & box C(i - 1);
end; # C

####################################################################################################
####################################################################################################
```